

Title:

## Metadata Schema

### Multi-cloud Execution-ware for Large-scale Optimised Data-Intensive Computing

H2020-ICT-2016-2017  
Leadership in Enabling and  
Industrial Technologies;  
Information and  
Communication  
Technologies

Grant Agreement No.:  
731664

Duration:  
1 December 2016 -  
30 November 2019

[www.melodic.cloud](http://www.melodic.cloud)

Deliverable reference:  
D2.4

Date:  
02 November 2017

Responsible partner:  
Institute of Communications  
and Computer Systems

Editor(s):  
Yiannis Verginadis

Author(s):  
Yiannis Verginadis,  
Ioannis Patiniotakis,  
Christos Halaris,  
Gregoris Mentzas,  
Kyriakos Kritikos,  
Keith Jeffery

Approved by:  
Ernst Gunnar Gran

ISBN number:  
N/A

Document URL:  
[http://www.melodic.cloud/deliverables/D2.4 Metadata Schema.pdf](http://www.melodic.cloud/deliverables/D2.4%20Metadata%20Schema.pdf)

Abstract:

This document introduces the initial definition of a vocabulary entitled *Melodic Metadata Schema*. This Schema aggregates a number of classes and properties that correspond to concepts used for describing requirements, constraints and offerings' characteristics in multi-cloud placement decisions. The description of such characteristics will constitute the background of the Melodic mechanisms for properly managing big data, optimising the placement of processing jobs and controlling access requests in multi-cloud environments. The Metadata Schema comprises the *Application Placement*, *Big Data* and *Context Aware Security* models that group a number of classes and properties to be used for defining where a certain big data application should be placed; what are the unique characteristics of the data artefacts that needs to be processed; and what are the contextual aspects that may be used for restricting the access to the sensitive data.

This deliverable also discusses the envisioned ways that the Metadata Schema can be used for extending the CAMEL language. Concepts from this Schema potentially affect the *Requirement*, *Metric*, *Scalability*, *Location*, *Provider* and *Security* sub-models of CAMEL.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

<b>Document</b>	
Period Covered	M1-10
Deliverable No.	D2.4
Deliverable Title	Metadata Schema
Editor(s)	Yiannis Verginadis
Author(s)	Yiannis Verginadis, Ioannis Patiniotakis, Christos Halaris, Gregoris Mentzas, Kyriakos Kritikos, Keith Jeffery
Reviewer(s)	Pawel Gora, Keith Jeffery
Work Package No.	2
Work Package Title	Architecture and Data Management
Lead Beneficiary	ICCS
Distribution	PU
Version	14.6
Draft/Final	Final
Total No. of Pages	101 + One Appendix

## Table of Contents

Metadata Schema .....	1
1 Introduction .....	6
1.1 Scope of the Document .....	6
1.2 Structure of the Document.....	9
2 Summary of Vocabularies and Ontologies Related to Data-aware Multi-cloud Computing .....	10
3 Metadata Schema for Data-aware Multi-cloud Computing.....	16
3.1 Overview .....	16
3.2 Application Placement Model .....	18
3.2.1 Application Placement Model Overview.....	18
3.2.2 Application Placement Model Details .....	19
3.3 Big Data Model.....	42
3.3.1 Big Data Model Overview.....	42
3.3.2 Big Data Model Details.....	44
3.4 Context Aware Security model .....	83
3.4.1 Context Aware Security model Overview.....	83
3.4.2 Context Aware Security model Details .....	84
4 CAMEL Updates based on Metadata Schema .....	90
5 Conclusions.....	96
References .....	98
Appendix – Metadata Schema Serialization .....	102

## List of Figures

Figure 1: The class diagram of the CAMEL metamodel denoting the sub-models that may be extended using the Metadata Schema.....	8
Figure 2: Part of the class diagram of the requirement package related to hardware, OS, image, and provider requirements (Rossini et al., 2015).....	10
Figure 3: Saloon Ontology (Quinton et al., 2012; Quinton et al., 2013).....	11
Figure 4: DICE Metamodel for big data intensive applications.....	13
Figure 5: Data Domains from the CSA Big Data Taxonomy (Murthy et al., 2014).....	14
Figure 6: PaaSWord Context Aware Security Model Overview (Verginadis et al., 2016) .....	15
Figure 7: Melodic's Metadata Schema Overview .....	16
Figure 8: Application Placement Model Overview.....	19
Figure 9: Application Placement Model's UML Class Diagram (1/2).....	40
Figure 10: Application Placement Model's UML Class Diagram (2/2).....	41
Figure 11: Big Data Model's Overview Diagram (1/3) .....	42
Figure 12: Big Data Model's Overview Diagram (2/3).....	43
Figure 13: Big Data Model's Overview Diagram (3/3).....	44
Figure 14: Big Data Model's UML Class Diagram (1/5) .....	78
Figure 15: Big Data Model's UML Class Diagram (2/5).....	79
Figure 16: Big Data Model's UML Class Diagram (3/5).....	80
Figure 17: Big Data Model's UML Class Diagram (4/5).....	81
Figure 18: Big Data Model's UML Class Diagram (5/5).....	82
Figure 19: Security Context Element -Melodic Extensions .....	83
Figure 20: Permission - Melodic Extensions.....	84
Figure 21: A snapshot of CAMEL focusing on its new concepts and extensions .....	92
Figure 22: CAMEL snippet showing how GPU capabilities and requirements can be specified.....	93
Figure 23: The transformation from Metadata Schema to CAMEL for Amazon AWS cloud locations.....	94
Figure 24: The bidirectional templating for metrics .....	95

## List of Tables

Table 1: Legend of the Overview and UML diagrams .....	18
Table 2: IaaS, PaaS, Provider Details .....	20
Table 3: Security Controls Details .....	34
Table 4: Big Data Aspects Details .....	45
Table 5: Data Location Details .....	52
Table 6: Data Management Details .....	55
Table 7: Data Domains Details .....	73
Table 8: Context Aware Security model Details .....	85

# 1 Introduction

This document reports on the initial design of Melodic's Metadata Schema for data-aware multi-cloud computing. Its objective is to aid the data management, access control, and data-aware application design for distributed and loosely-coupled multi-cloud applications. This objective is addressed by introducing terminology and vocabulary aspects of metadata that will be mainly used for extending the domain specific language (DSL), i.e. the Cloud Application Modelling and Execution Language (CAMEL) (Kritikos et al., 2014; Rossini et al., 2015), that Melodic will use for describing big-data applications, their requirements and the available offerings. We note that the Melodic's Metadata Schema provides a thesaurus structure that describes entities and their interrelations. Specifically, the schema hierarchically structures, into a vocabulary, all the concepts (represented as lexical terms) that are relevant for describing cloud application requirements, big data aspects and characteristics (with respect to the input and output of these applications) and the offered cloud infrastructure capabilities for discovering optimised multi-clouds placement opportunities. Moreover, this Metadata Schema will encapsulate all the necessary concepts for enabling the context-aware authorization functions that the Melodic platform envisions to support.

## 1.1 Scope of the Document

One of the first decisions of the Melodic consortium was to use this vocabulary in order to extend the Cloud Application Modelling and Execution Language (CAMEL) (Kritikos et al., 2014; Rossini et al., 2015), developed in the frame of the PaaSage<sup>1</sup> project. CAMEL enables the specification of multiple aspects of multi-cloud applications (i.e., applications deployed across multiple private, public, or hybrid cloud infrastructures), facilitating the optimised application placement and adaptation over multiple cloud infrastructures. This approach follows the model-driven engineering (MDE) paradigm (Frankel, 2003) that enables the modelling abstraction from the implementation details of heterogeneous cloud services. This also enables the development of appropriate mechanisms that allow both direct and programmatic manipulation of design and runtime models in order to facilitate the efficient matchmaking between cloud applications' requirements and the available multi-cloud offerings. Among others, CAMEL introduces or builds on top of various sub-models in order to support the specification of cloud application requirements (e.g. Hardware, OS & Image and Provider Requirements, Location requirements, Security requirements, Scalability requirements/rules, Service Level Objectives (SLOs)). Based on these sub-models an application developer is able to describe its application requirements that

---

<sup>1</sup> <https://paasage.ercim.eu/>

will drive the multi-cloud placement process (see for example List 1 that captures a part of a cloud application specification in CAMEL that requires that the certain application should be placed only on VMs located in Germany and with a number of cores between 8 and 32 and RAM between 4 and 8 MB). The main issue with the current status of CAMEL is that any concept (e.g. number of cores, RAM, location etc.) that can be used in a requirement specification has been statically predefined, while the process for extending such a vocabulary can be proven cumbersome. This is an issue especially for the Melodic platform where the use of CAMEL should be accompanied by the capability to provide data-awareness in the cloud application specifications. For example, the optimal placement of a cloud application that performs batch processing over petabytes of data, is likely to be different from the optimal placement of an application that conducts real-time processing over data streams with a velocity of several gigabytes per second). Thus, big data related requirements need to be supported by an extended CAMEL by incorporating a number of additional vocabulary terms representing concepts according to the Melodic adopter’s needs (e.g. VM requirements based on GPU offerings). Melodic’s Metadata Schema will constitute the medium for modelling any concept necessary for expanding the CAMEL’s expressivity with respect to cloud application requirements specifications and offerings descriptions, respectively (i.e. CAMEL’s requirement and provider sub-models).

<pre> requirement model <b>ScalarmRequirement</b> {   quantitative hardware <b>CoreIntensive</b> {     core: 8..32     ram: 4096..8192   }   os <b>Ubuntu</b> {os: 'Ubuntu' 64os}   location requirement <b>GermanyReq</b> {     locations [<b>ScalarmLocation.DE</b>]   } </pre> <p><i>List 1: Requirement Model Example in CAMEL</i></p>	<pre> location model <b>ScalarmLocation</b> {   region EU {     name: 'Europe'   }   country DE {     name: 'Germany'     parent regions [<b>ScalarmLocation.EU</b>]   } } </pre> <p><i>List 2: Location Model Example in CAMEL</i></p>
--	---

Furthermore, this Metadata Schema will affect and extend the available concepts representing raw metrics (i.e. attributes measurable whose values can be obtained from system sensors; e.g. current CPU usage), along with the variables (i.e. attributes for which any of the solver mechanisms (Zahid et al., 2017) are the sensors that assign possible values) used for describing a utility function (capturing preferences) or declaring constraints, thus bounding the application placement problem. In Figure 1, we denote with red circles the classes of the CAMEL metamodel

that may be extended using the Metadata Schema vocabulary. Based on this, important mechanisms of the Melodic Upperware, as they were mentioned in the initial system specification of the Melodic multi-cloud middleware platform mechanisms (Zahid et al., 2017), will be affected by the concepts modelled in the Metadata Schema. Specifically, the Solvers may use several of the classes of the Metadata Schema as these will be introduced in the utility function to be resolved, while the Metasolver or some other software components can use them as criteria for comparing local optimal solutions coming from the solvers. In addition, these classes can be used by the adapter as criteria for evaluating a new deployment plan that adapts a current topology. Thus the Schema should be amendable in order to allow for future incorporation of needs of Melodic adopters. For this purpose, we note that the Metadata Schema reported in this document constitutes only a first iteration and will be adjusted according to the needs of each of the Melodic pilot partners using a dedicated editor developed by the Melodic project.

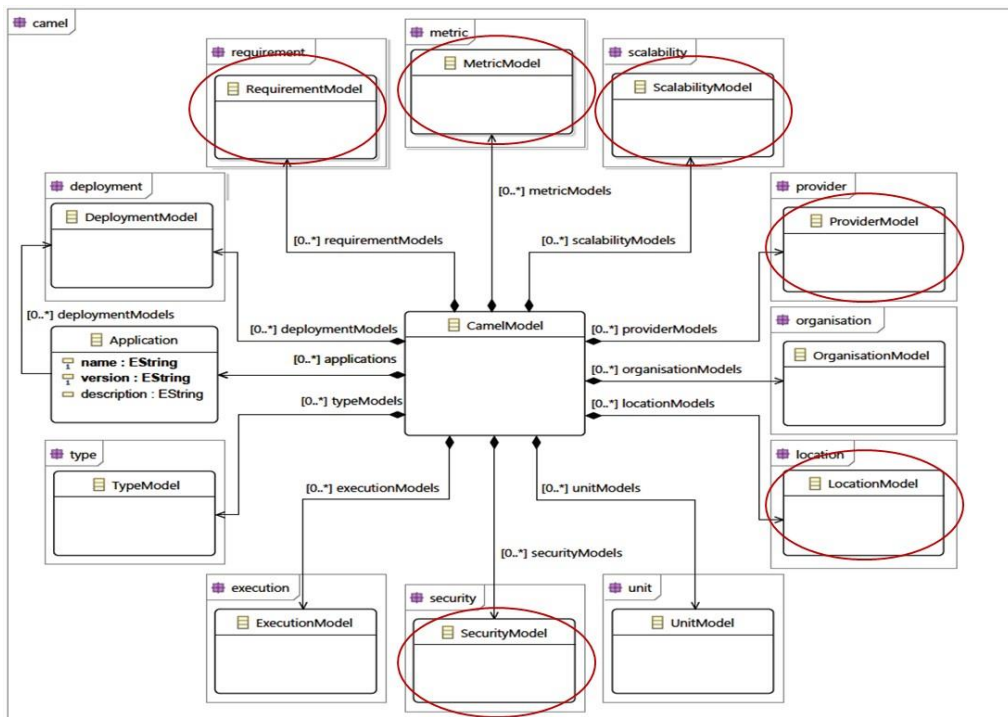


Figure 1: The class diagram of the CAMEL metamodel denoting the sub-models that may be extended using the Metadata Schema

Summarizing the introduction of Melodic’s Metadata Schema brings the following advantages:

- Formally and graphically declare, in a vocabulary, all the necessary terms that describe concepts to be used for comparing deployment alternatives
- Add data aspects in CAMEL without hard coding any concepts



- Provide a unified way for stating the vocabulary terms' importance per each case (e.g. Meta-Solver or adapter)
- Add concepts to be used by the Melodic's authorization engine
- Support model extensibility by easily incorporating other vocabularies
- Support model reusability
  - e.g. the same location model should be used for any application modelled in the same organization (e.g. see List 2)

## 1.2 Structure of the Document

This deliverable begins with an introductory chapter (Chapter 1) that discusses the main idea around the Metadata Schema along with its role in the Melodic platform. Chapter 2 aggregates and discusses all the relevant vocabularies or ontologies that mention concepts related to data-aware multi-cloud computing and notes which aspects of them are re-used or extended in terms of the Melodic Metadata Schema. In Chapter 3, we present the main aspects and details of the first iteration of the Melodic Metadata Schema that includes the Application Placement, Big Data-Aware and Context-Aware Security models. In Chapter 4, we sketch the anticipated CAMEL updates based on this Metadata Schema. Last, in Chapter 5, we conclude the discussion on this Schema. We note that an example of the XMI serialization of the current version of the Schema is provided in Appendix I.

## 2 Summary of Vocabularies and Ontologies Related to Data-aware Multi-cloud Computing

In this chapter, we discuss relevant vocabularies and ontologies already used in CAMEL or in other big data related projects. We focus only on efforts that introduce relevant concepts or hierarchies, valuable for constructing a generic schema that may serve as a background vocabulary to be used for defining the critical concepts that will drive the placement and reconfiguration of big data applications over multi-cloud resources.

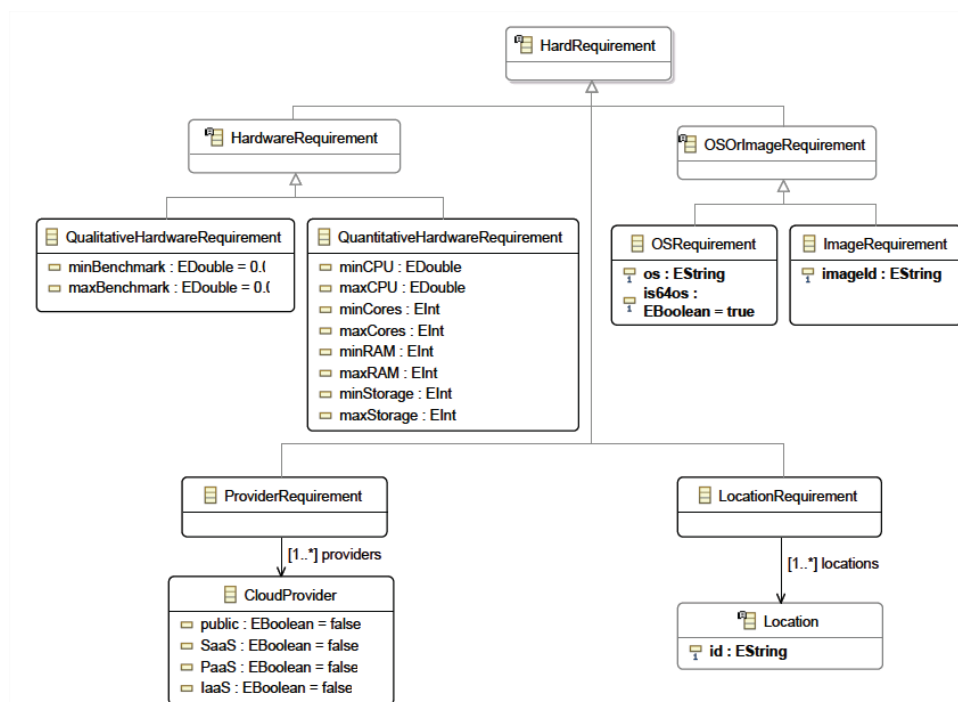


Figure 2: Part of the class diagram of the requirement package related to hardware, OS, image, and provider requirements (Rossini et al., 2015)

From a focused state of the art analysis, we have detected a number of efforts that introduce definitions and taxonomies of relevant concepts. For example, the work of Ranjan et al. (2015) presents an overview of cloud resource orchestration programming issues, where several IaaS and PaaS concepts, but also data persistence and distributed ML appliances, have been defined. In another interesting work (Höfer & Karagiannis, 2011) that analyses the available cloud computing services and identifies some of their main characteristics, the authors propose a tree-structured taxonomy. Its purpose is to enable quick classifications and comparisons among different cloud computing services by starting from and elaborating on the IaaS/PaaS/SaaS classification. Youseff et al. (2008) proposed a unified ontology of cloud computing, defining concepts distinguished in five layers, with three constituents to the cloud infrastructure layer. This work mainly discusses the inter-dependency and composability between the different layers in the cloud (i.e. SaaS, PaaS, IaaS, Data-Storage as a Service (DaaS), and Communication as a

Service (CaaS)). A similar approach is also introduced (Kang & Sim, 2011) that presents a search engine for cloud computing system and introduces the CO-1 and CO-2 ontologies to semantically define the relationship among cloud services. It is used for determining the similarity among cloud services using three types of reasoning (i.e. concept similarity reasoning, object property similarity reasoning, and datatype property similarity reasoning). Based on these works we reuse and extend some of the definitions provided, as detailed in Chapter 3.

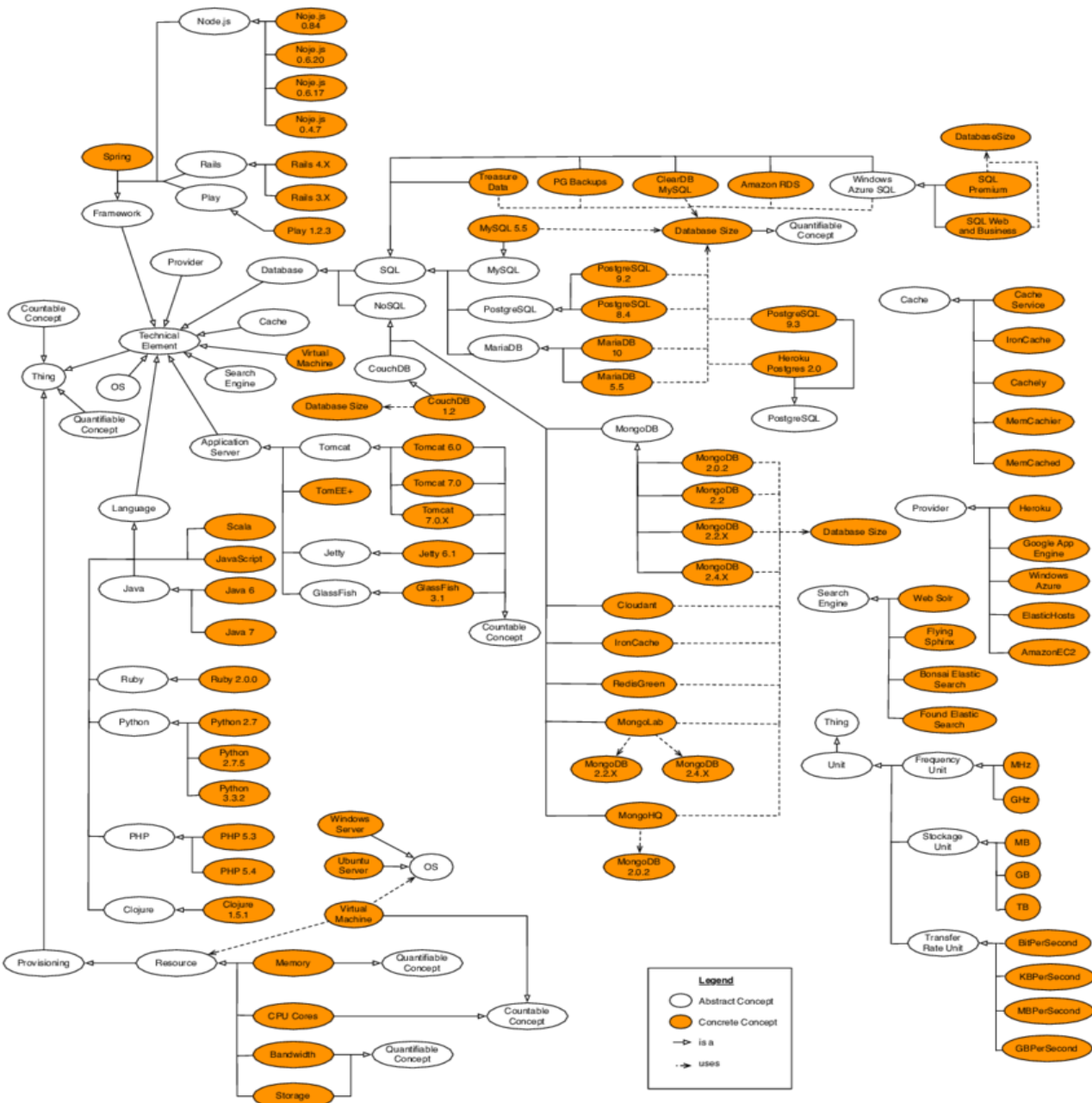


Figure 3: Saloon Ontology (Quinton et al, 2012; Quinton et al, 2013)

In the rest of this chapter, we focus on research efforts that have a bigger impact on the Metadata Schema. We start with CAMEL that, as already mentioned, will be reused and extended for the

Melodic purposes. Nevertheless, CAMEL already includes a number of concepts (through re-used domain specific languages (DSLs) or new sub-models) that should constitute the starting point for designing the Metadata Schema. CAMEL is a core modelling and execution language that enables the specification of multiple aspects of cross-cloud applications. One of the most relevant sub-models of CAMEL refers to the requirement package of the CAMEL metamodel, which has been developed to enable the specification of requirements for cross-cloud applications (Rossini et al., 2015). As can be seen in Figure 2, a number of concepts are introduced as hardware, OS, provider or location requirements. These concepts are also introduced in the Application Placement model of the Metadata Schema.

A second part of CAMEL that fused the Metadata Schema initial iteration with important concepts, is the provider package. This package is based on the Saloon ontology (Quinton et al., 2012; Quinton et al., 2013). Saloon (Figure 3) is a tool-supported DSL for specifying the features of cloud providers and matching them with requirements by leveraging feature models (Benavides et al., 2010) and ontologies (Gruber, 1993).

Besides the CAMEL's Requirement and Provider model, from where the Metadata Schema re-uses (and enriches) concepts, the following parts of CAMEL will be fused with concepts from the Metadata Schema:

- the scalability and metric packages, which are based on the Scalability Rule Language (SRL) (Kritikos et al., 2014) where for example instances of the class Metric will be based on the Metadata concepts
- the location package, which is not based on existing DSLs and has been developed to enable the specification of locations where the hierarchy of physical, network and cloud locations from the Metadata Schema can be used for extending it (Rossini et al., 2015)
- the security package, which is also not based on existing DSLs and has been developed to enable the specification of security aspects of cross-cloud applications. (Kritikos & Massonet, 2016)

Another relevant effort is the DICE<sup>2</sup> project that focuses on quality-driven development of big data applications. DICE offers a UML profile and tools that assist software designers reasoning about reliability, safety and efficiency of data-intensive applications. The DICE methodology covers quality assessment, architecture enhancement, continuous testing and agile delivery, relying on principles of the emerging DevOps paradigm (Gómez et al., 2016). Specifically, it has introduced a Metamodel (Figure 4) for describing aspects of big data intensive applications that was built on top of:

---

<sup>2</sup> <http://www.dice-h2020.eu/>

- MARTE (Modelling and Analysis of Real-Time and Embedded systems) model, which provides support for the specification, design, quantitative evaluation, and verification and validation of software systems (OMG, 2011).
- DAM (Dependability Analysis and Modelling Profile) model, which provides support for the dependability modelling and analysis of software systems (Bernardi et al., 2013).

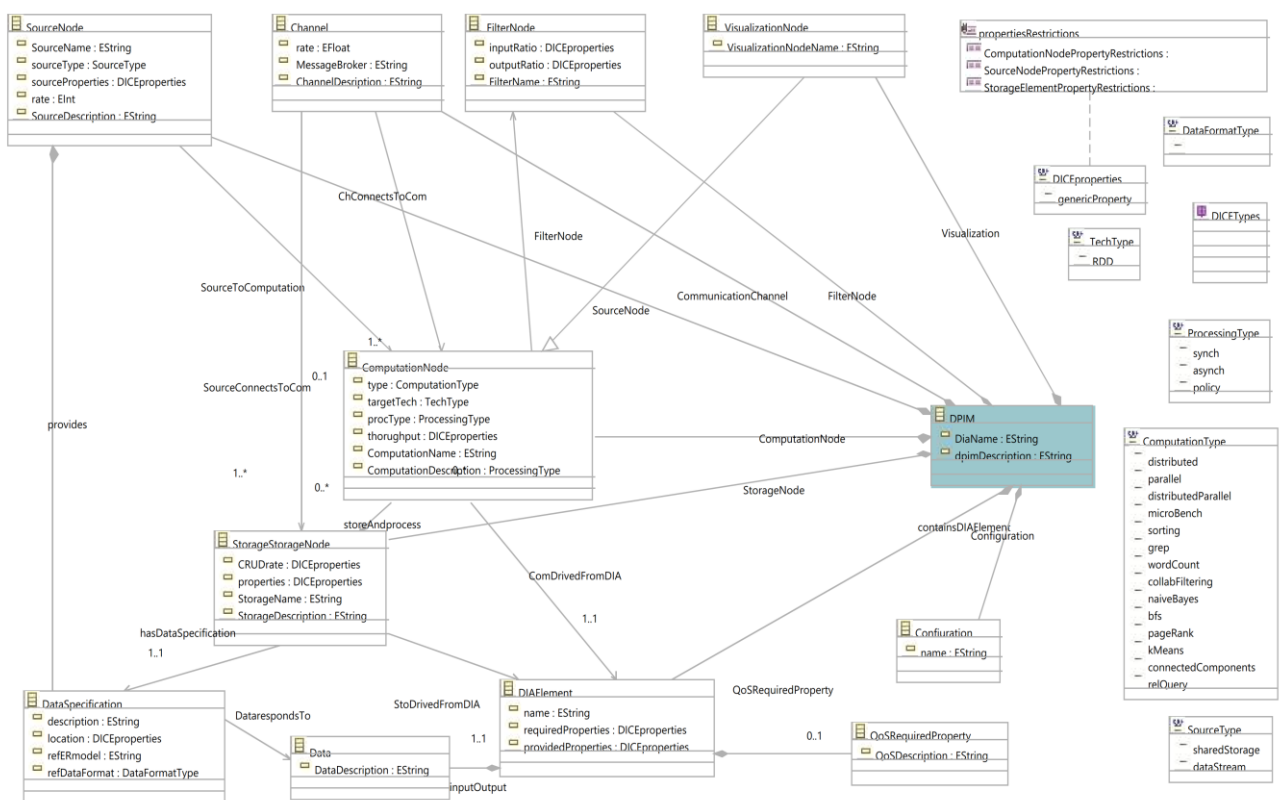


Figure 4: DICE Metamodel for big data intensive applications

With respect to reusing concepts introduced by DICE in the Melodic Metadata Schema, several classes and properties are considered relevant (e.g. computation and storage nodes, computation and processing types etc.). Nevertheless, there is no direct support for expressing data location and big data related properties such as volume, transfer rates or even aspects of the operations that transfer data between cloud resources. Such concepts are covered in the Melodic metadata Schema (discussed in Chapter 3).

An additional relevant work (Kleppmann, 2017) tried to aggregate the “Big ideas behind reliable, scalable, and maintainable systems” focusing on data-intensive applications with the aim to enhance the understanding of the diverse and fast-changing landscape of technologies for processing and storing big data. Several details on map-reduce (Chu et. Al, 2007) and graph-like

data models aspects, the distribution and replication of data techniques (for high availability, scalability, latency) and specifics on data partitioning, were identified as relevant and reusable for the Metadata Schema. Moreover, the relevant attributes and characteristics (e.g. throughput) depending on the data processing type (e.g. batch processing) were also considered. Although this work did not introduce any distinct taxonomies, the definitions and analyses of the major characteristics and challenges concerning the data-intensive applications were very beneficial with respect to a number of definitions used for the Melodic vocabulary.

The work on the Metadata Schema considered or directly reused concepts and hierarchies introduced by the Cloud Security Alliance (CSA) with respect to a big data taxonomy (Murthy et al., 2014). That specific work, proposed a six-dimensional taxonomy with the aim of assisting decision makers to navigate through the plethora of choices in compute and storage infrastructures as well as data analytics methods, and security and privacy frameworks (Murthy et al., 2014). For example, in the Metadata Schema we reused and slightly extended the data domains taxonomy introduced by CSA which can be found in Figure 5. In addition, we considered latency requirements (e.g. (near) real-time, batch), compute (e.g. batch, streaming) and storage infrastructures (e.g. SQL, NoSQL, NewSQL) along with processing complexity aspects.

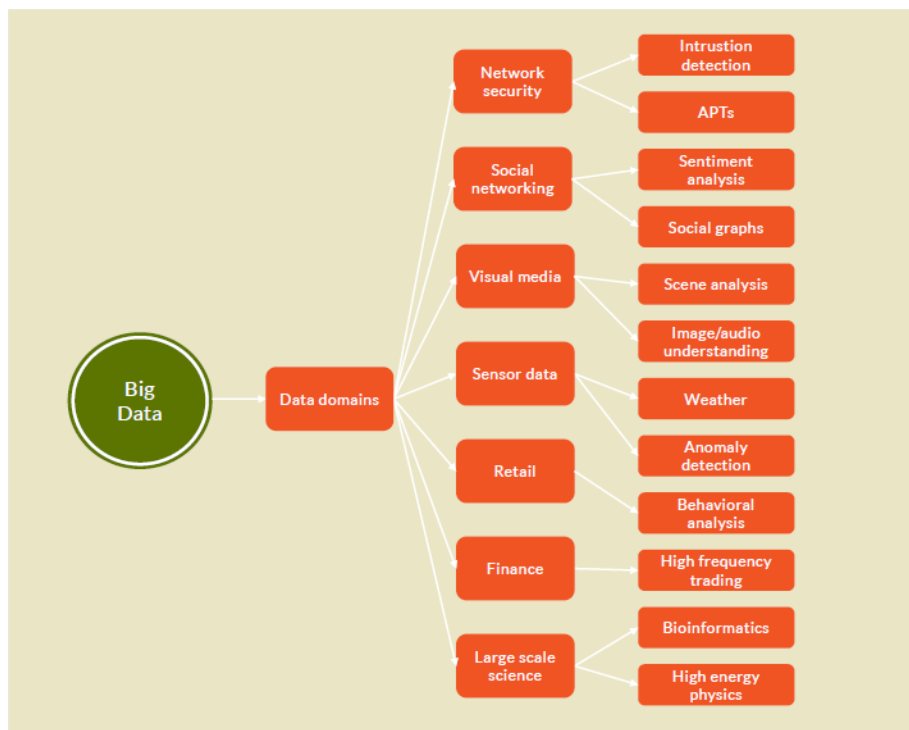


Figure 5: Data Domains from the CSA Big Data Taxonomy (Murthy et al., 2014)

Another important ontology that we have considered for the Metadata Schema is the Context Aware Security Model of the PaaSWord project (Verginadis et al., 2016; Veloudis et al., 2016). In Figure 6, the reader may find an overview of the Context Aware Security model as it has been

described in the Paasword project (Verginadis et al., 2016). This semantic model is reused in terms of a hierarchical structure of classes and properties that provides a complete set of concepts introduced into the Melodic vocabulary in order to address the description of context associated to processing or access control decisions.

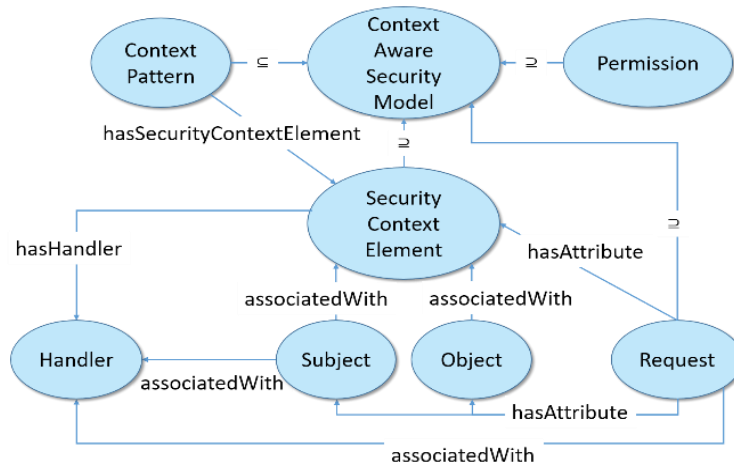


Figure 6. PaaSword Context Aware Security Model Overview (Verginadis et al., 2016)



### 3 Metadata Schema for Data-aware Multi-cloud Computing

#### 3.1 Overview

In this chapter we introduce the Melodic Metadata Schema that comprises the following model facets:

- Application Placement Model
- Big Data Model
- Context Aware Security model

The several different facets of the Melodic Metadata Schema are analysed below, while a bird’s eye view of the schema can be found in Figure 7. For each of the facets their main top-level concepts are also depicted while explained in the sections below. For the representation of a comprehensible overview of the Melodic Metadata Schema, we used a free, HTML5-compliant mind mapping webapp with cloud support. We note that the detailed mind map for an easier walkthrough of the Schema’s main aspects can be found here:

[http://melodic.cloud/assets/images/MELODIC\\_Model\\_vFinal.png](http://melodic.cloud/assets/images/MELODIC_Model_vFinal.png)

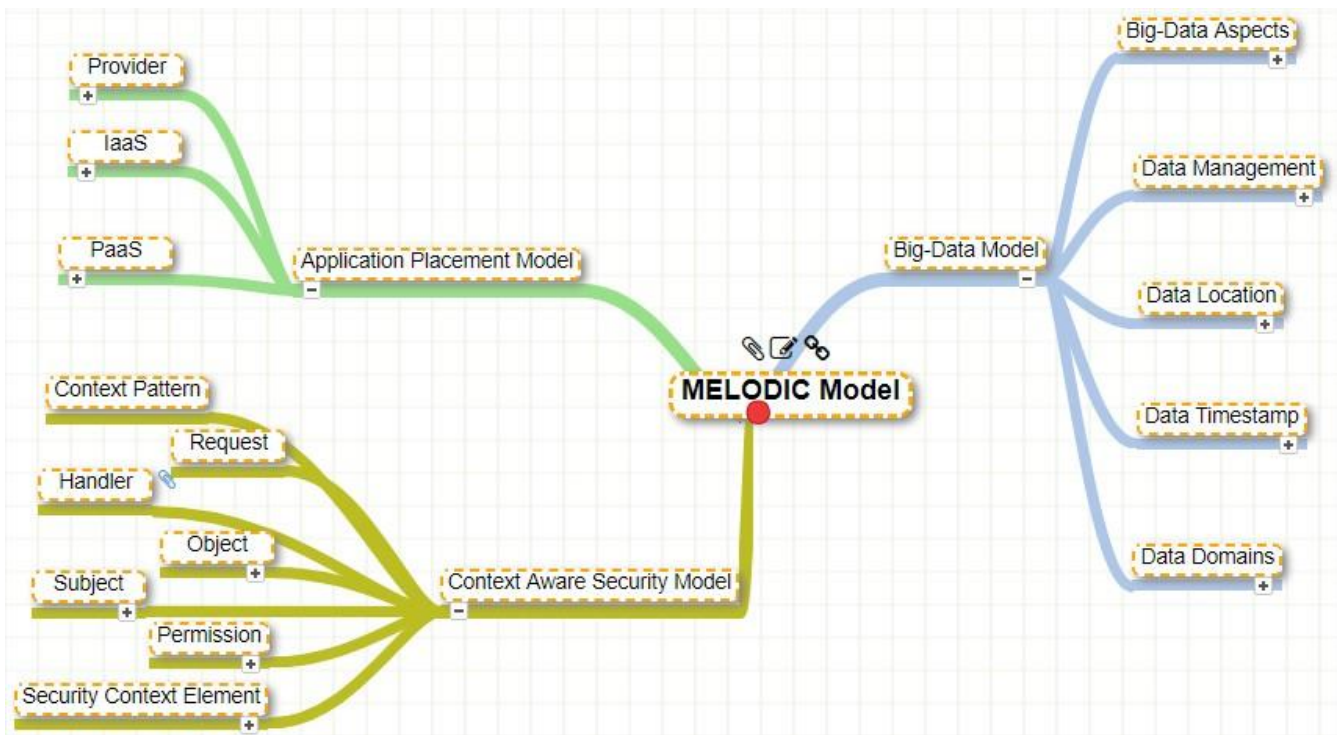


Figure 7: Melodic’s Metadata Schema Overview

The first model facet of the Schema is the Application Placement Model, which provides a hierarchical structure over a number of concepts and properties that can be used either for describing cloud application placement requirements, constraints and preferences, or for



describing the available cloud offerings, mainly at the IaaS and PaaS levels. This includes concepts that reveal processing (e.g. CPU), storage (e.g. Capacity), network (e.g. Bandwidth) as well as hypervisor characteristics or capabilities at IaaS level. At a PaaS level, we include concepts that characterise the available or required platform type, environment (e.g. OS) as well as the security controls (e.g. Data Sanitization) that it currently supports. As mentioned in the previous chapter, for deriving this set of classes and properties of this facet, the main vocabularies and/or ontologies that we reused and extended were the Saloon ontology (Quinton et al., 2012; Quinton et al., 2013), and CAMEL (Rossini et al., 2015).



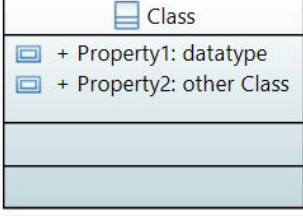

The second model facet of the Schema is the Big Data Model for multi-cloud management. This model provides a hierarchical structure over a number of concepts and properties that can be used for describing characteristics of data to be processed, that should be considered during application placement or cloud reconfiguration decisions. As mentioned in Chapter 2, for deriving this model facet we mainly built on and extended several vocabularies like: DICE (Gómez et al., 2016), the work of (Kleppmann, 2017) and the CSA Big Data Taxonomy (Murthy et al., 2014). Nevertheless, to the best of our knowledge, this is the first systematic effort that tries to capture all the different data-related aspects that are important for data intensive applications. Thus, this model facet reveals big data aspects (e.g. Volume, Velocity, Quality etc.), data management details (e.g. Acquisition, Data Storage, Processing etc.), data location and timestamp along with the relevant data domains (e.g. Finance, Social Networking etc.), that characterize the big data to be processed in multi-cloud environments.

The last model facet of the Schema is the Context Aware Security model. This model aggregates a number of concepts and properties for describing and enforcing context-aware access control policies. This part corresponds to the Context-aware model developed by ICCS in terms of the PaaSword project (Verginadis et al., 2016; Veloudis et al., 2016), extending it with a number of concepts that consider the infrastructural requirements and available offerings in the Melodic application scenarios. Based on the security context elements or the context patterns described, the Melodic consortium will implement an authorization engine that will enhance the access control to sensitive big data, managed by Melodic-enabled multi-cloud applications.

We note that the Melodic's Metadata Schema provides a thesaurus structure that describes entities and their interrelations. As the work with the Melodic Upperware progresses, a need might be identified to specifically define a set of valid values (terms) per each concept introduced, to be injected in the CAMEL sub-models. In this case, these allowed values will be included in the Metadata Schema as some class instances (to be introduced by a dedicated editor). These instances may bound the CAMEL web-editor to the valid values that the Melodic adopter would wish to restrict. It is also important to mention that this document reports on the first iteration of the Melodic's Metadata Schema, for which a dedicated editor will be developed as part of the Melodic project so the Schema can be easily updated by any of the Melodic adopters.

In Table 1, we provide the legend for the Overview and the UML Class diagrams that are used in the following sub-sections.

Table 1: Legend of the Overview and UML diagrams

	Class denoting a concept defined in the <i>Melodic Metadata Schema</i> . It might be a subclass of another Class. (Appears in Overview diagrams)
	Captures a <i>subClassOf</i> relation. The arrowhead indicates the parent class. (Appears in Overview & UML diagrams)
	Class defined in the Melodic Metadata Schema. It might be a subclass of another class and it may include two types of properties: i) data properties that have as range a datatype and ii) object properties that have as range another class of the <i>Melodic Metadata</i> . (Appears in UML class diagrams)
	Captures an object property relation between two classes (domain and range classes). The arrowhead indicates the range class. (Appears in Overview & UML diagrams)

## 3.2 Application Placement Model

### 3.2.1 Application Placement Model Overview

In Figure 8, an overview of the core classes and sub-classes of the Application Placement Model is provided.

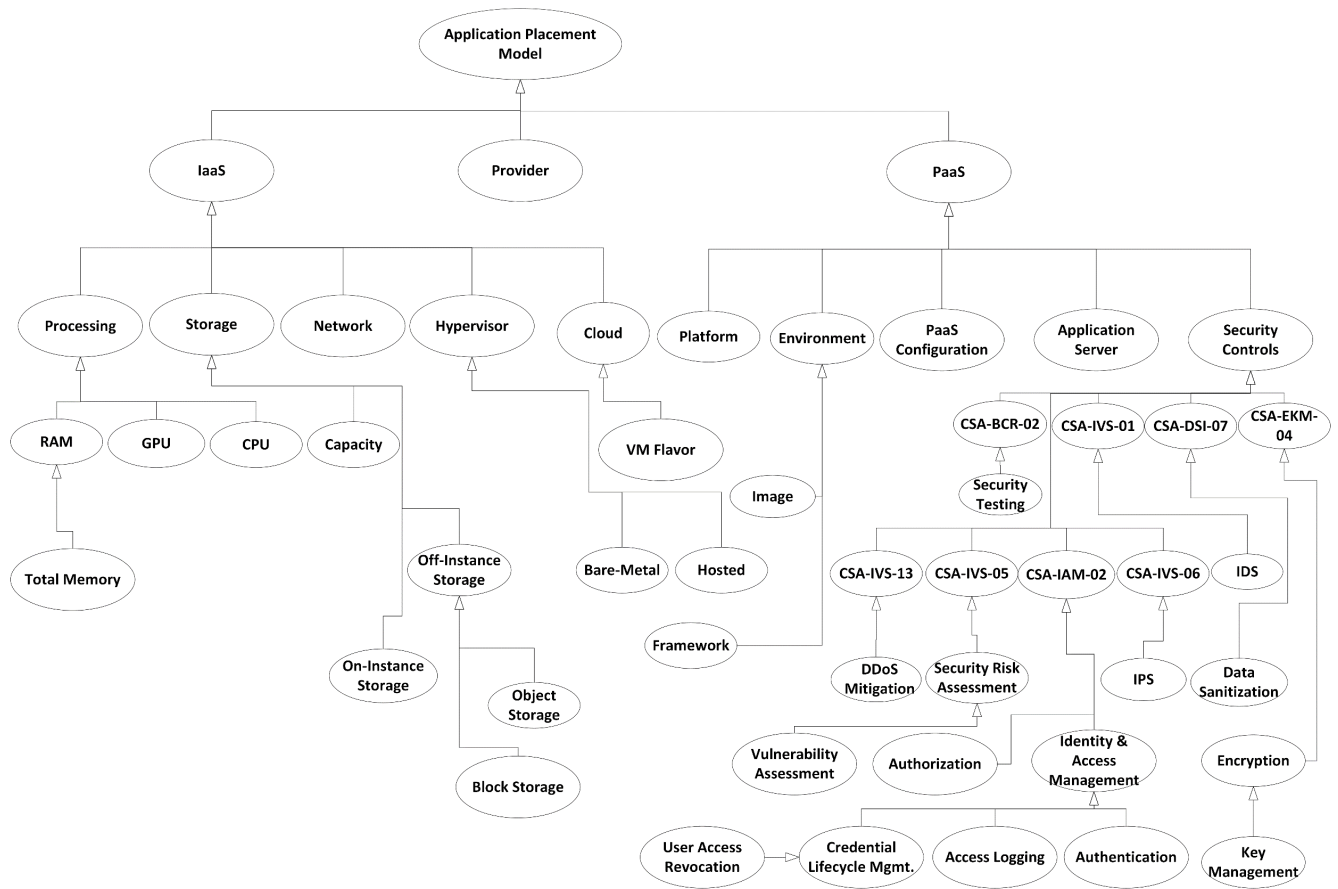


Figure 8: Application Placement Model Overview

### 3.2.2 Application Placement Model Details

The Application Placement model refers to the following top-level concepts:

- IaaS
  - Processing
  - Storage
  - Network
  - Cloud
  - Hypervisor
- Provider
- PaaS
  - Platform
  - Application Server

- Environment
- PaaS Configuration
- Security Controls

For each of these top-level core classes, we provide their respective subclasses, their main properties along with their descriptions. This includes the following details table where each concept is explained. In Table 2 and Table 3 the details of the Application Placement model are presented. We used two separate tables for improving the involved concepts readability.

Table 2: IaaS, PaaS, Provider Details

Class Taxonomy Levels				Properties	Description
IaaS					This class encapsulates all the attributes related to cloud infrastructural resources that are required and offered for deploying Melodic-enabled applications. It reuses and extends the requirement model of CAMEL (Rossini et al., 2015).
				<i>refersToVM</i>	This property clarifies that a certain IaaS resource (required or offered) is a virtual machine.
				<i>refersToRack</i>	This property clarifies that a certain IaaS resource (required or offered) corresponds to a specific rack of a datacentre.
				<i>Supports RequestsPer Second</i>	This property may associate an IaaS resource with an integer that expresses the volume of requests that it can support.
				<i>hasVMCost</i>	This property associates the IaaS class with a value expressed in floating-point format (float) denoting the usage cost of a certain virtualised resource.
				<i>hasBare MetalCost</i>	This property associates the IaaS class with a value expressed in floating-point format (float) denoting the

					usage cost of a certain bare metal resource.
				<i>has Availability</i>	This property may associate an IaaS resource with a float that expresses the expected uptime of the resource.
				<i>hasCloud Location</i>	This object property associates the IaaS class with the Cloud Location Class (of the Big Data Model) for expressing the network or physical location of the virtualised resource.
				<i>hasCloud Provider</i>	This object property associates the IaaS class with the Provider Class (of the Application Placement Model) for expressing characteristics and identity of the resource provider.
	Processing				This class involves any infrastructural feature bound to the processing capability of virtualised resources.
		RAM			This class corresponds to the memory capabilities of a virtualised resource.
				<i>hasFree Memory</i>	This property associates the RAM class with a value expressed in double-precision floating-point format (double) that denotes the amount of unused memory currently available by the virtualised resource.
				<i>hasUsed Memory</i>	This property associates the RAM class with a value expressed in double format that denotes the amount of used memory in the virtualised resource.

				<i>has Manufacturer</i>	This property associates the RAM class with a string that denotes the producer of the hardware.
			Total Memory		This subclass captures the desired or offered value of the virtualised storage dedicated for frequent program instructions.
				<i>hasMin</i>	This property associates the Total Memory class with an integer that represents the least amount of memory capacity required or offered.
				<i>hasMax</i>	This property associates the Total Memory class with an integer that represents the largest amount of memory capacity required or offered.
				<i>hasUnit</i>	This property associates the Total Memory class with a string that represents the measurement module of the memory capacity.
		GPU			This class refers to IaaS resources that use graphics processing units (GPUs), i.e. specialized electronic circuits initially designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer.
				<i>hasStart UsageDate</i>	This property denotes the date when a certain GPU began to operate. It can be used as an attribute that reveals how new the processing units used by a certain IaaS resource are.

			<i>has Manufacturer</i>	This property expresses as a string the manufacturer of the processing unit.
			<i>hasGPU Utilization</i>	This property associates the GPU class with a double that represents the current percentage of use for a certain processing unit.
			<i>hasMFLOPs</i>	This property associates the GPU class with an integer that represents the capability for mega floating-point operations per second, which is a common measure of processing speed.
			<i>hasPEperCUs</i>	This property expresses with an integer the number of processing elements per compute units that a certain GPU offers.
			<i>hasWarpSize</i>	This property expresses as an integer the number of threads supported to coalesce memory access and instruction dispatch.
			<i>hasMax Concurrent Workgroups</i>	This property denotes an integer that represents the maximum work-groups that may be simultaneously executed on compute units supported by a certain GPU.
			<i>hasMin Numberof Cores</i>	This property denotes an integer that captures the minimum number of GPU cores available or requested.
			<i>hasMax Numberof Cores</i>	This property denotes an integer that captures the maximum number of GPU cores available or requested.
			<i>hasClock Speed</i>	This property captures the GPU operating speed expressed in

					cycles per second. It associates the GPU class with an integer value.
		CPU			This class refers to IaaS resources that use Central Processing Units (CPUs) for carrying out software instructions that specify the basic arithmetic, logical, control and input/output (I/O) operations.
				<i>hasCPU Utilization</i>	This property associates the CPU class with a double that represents the current percentage of use for a certain processing unit.
				<i>hasMIPs</i>	This property associates the CPU class with an integer that expresses million instructions per second as a measure of processing speed supported by a certain IaaS resource.
				<i>hasMFLOPs</i>	This property associates the CPU class with an integer that represents the capability for mega floating-point operations per second.
				<i>has Manufacturer</i>	This property expresses as a string the manufacturer of the certain processing unit.
				<i>hasMin Numberof Cores</i>	This property denotes an integer that captures the minimum number of CPU cores available or requested.
				<i>hasMax Numberof Cores</i>	This property denotes an integer that captures the maximum number of CPU cores available or requested.
				<i>hasFrequency</i>	This property captures the CPU performance as it specifies the



					operating frequency of the CPU cores, expressed in cycles per second. It associates the CPU class with an integer value.
	Storage				This class describes the ephemeral or persistent storing capabilities that are required or offered by a certain IaaS resource.
				<i>hasWriteCost</i>	This property associates the Storage class with a float that represents the cost for accumulating data artefacts in a certain IaaS resource.
				<i>hasReadCost</i>	This property associates the Storage class with a float that represents the cost for retrieving data artefacts from a certain IaaS resource.
				<i>hasDiskUsage</i>	This property denotes a float that represents the percentage of storing space used for persisting data artefacts.
				<i>hasWrite Throughput</i>	This property associates the Storage class with a float that represents the volume of data artefacts that can be stored in a certain IaaS resource.
				<i>hasRead Throughput</i>	This property associates the Storage class with a float that represents the volume of data artefacts that can be retrieved a certain IaaS resource.
				<i>hasStorage Location</i>	This object property associates the Storage class with the Cloud Location class of the Big Data Model for registering the location of an IaaS resource with storing capabilities.

				<i>hasStorage CostFunction</i>	This property associates the Storage class with a string that refers to a function that provides an accurate calculation of the expected cost for storing data artefacts.
				<i>isEnergy Efficient</i>	This property associates the Storage class with a boolean value that reveals the level of efficacy with respect to how much energy is needed for exploiting the storage capabilities of a resource.
				<i>hasSolid StateDrive</i>	This property associates the Storage class with a boolean value that denotes whether or not a certain resource that provides storage capabilities, is using a type of non-volatile storage that is able to store and retrieve data artefacts using only electronic circuits (without any involvement of moving mechanical parts). Based on this property a system may infer the energy efficiency and throughputs supported by a certain resource.
				<i>isPersistent</i>	This property associates the Storage class with a boolean value that states the ephemeral or persistent nature of storing capabilities offered by a certain IaaS resource.
		Capacity			This subclass is used for stating the size of storage space requested or offered by a certain IaaS resource.
				<i>hasMin</i>	This property denotes an integer that captures the

					minimum size of storage available or requested.
				<i>hasMax</i>	This property denotes an integer that captures the maximum size of storage available or requested.
				<i>hasUnit</i>	This property refers to a string that states the measurement unit used for declaring the storage capacity of a certain resource (e.g. GBs).
		On-Instance Storage			This subclass refers to ephemeral storage capabilities requested or offered by a virtualised resource. Essentially the storage capabilities described with this class last until the certain resource is decommissioned.
		Off-Instance Storage			This subclass refers to persistent storage capabilities requested or offered by a virtualised resource.
			Object Storage		This subclass refers to a type of persistent storage where data are stored as objects encapsulating the data artefacts, their metadata and a globally unique identifier.
			Block Storage		This subclass refers to a type of persistent storage where data are stored into evenly sized blocks, each with its own unique address.
	Network				This class refers to the network related aspects that bound the operation of an offered or a requested IaaS resource.
				<i>has Bandwidth</i>	This property associates the Network class with a float value

					that states the maximum throughput of a logical or physical communication path. It corresponds to the number of bits that can be conveyed per unit of time.
				<i>hasSubnet</i>	This property corresponds to a string value that represent the subnet of an IaaS resource.
				<i>hasAvailabilityZone</i>	This object property associates the Network class to the Availability Zone class of the Big Data Model for denoting isolated locations used to make network resources available.
				<i>hasNetworkCost</i>	This property denotes a float value that expresses the cost for exploiting the network resources.
				<i>hasIPType</i>	This property is used to express the required or offered type of internet protocol with respect to its access availability. It associates the Network class with a string.
				<i>hasPublicIP</i>	This property corresponds to a string value that represents the public IP of an IaaS resource.
				<i>hasPrivateIP</i>	This property corresponds to a string value that represents the private IP of an IaaS resource.
				<i>hasVersion</i>	This property is used to express the required or offered IP version (e.g. IPv4, IPv6). It associates the Network class with a string.
	Cloud				This class groups the characteristics of virtualised resources for easier reference and use.

				<i>hasName</i>	This property corresponds to a string value that denotes the name of a cloud offering for easy referencing.
				<i>hasID</i>	This property corresponds to an integer that denotes the identifier of a cloud offering for easy referencing.
				<i>isDedicated</i>	This property associates the Cloud class with a boolean value that denotes whether or not the host of the offered resource is dedicated to only one user or more.
				<i>isPublic</i>	This property associates the Cloud class with a boolean value that denotes whether or not the offered or requested cloud resource is public.
		VM Flavour			This subclass denotes virtual hardware templates required or offered. This indirectly defines important IaaS level values (e.g. RAM, disk, number of cores etc.) of the offered or required IaaS resources.
				<i>hasName</i>	This property corresponds to a string value that denotes the name of a VM flavour for easy referencing.
				<i>hasCPU</i>	This object property associates the Cloud class to the CPU class of the Application Placement model for denoting the CPU power of the IaaS resource.
				<i>hasGPU</i>	This object property associates the Cloud class to the GPU class of the Application Placement model for denoting the GPU power of the IaaS resource.

				<i>hasRAM</i>	This object property associates the Cloud class to the RAM class of the Application Placement model for denoting the available RAM of the IaaS resource.
				<i>hasStorage</i>	This object property associates the Cloud class to the CPU class of the Application Placement model for denoting the storage capability of the IaaS resource.
	Hypervisor				This class is used to express the characteristics of the used hypervisor software, firmware or hardware for creating and commissioning virtual machines.
				<i>Supports Virtualization Format</i>	This property associates the Hypervisor class with a string that denotes the format used for describing a virtual machine (e.g. Open Virtualization Format).
		Bare-Metal			This subclass captures the characteristics of hypervisors that operate directly on the hardware for hosting guest operating systems.
		Hosted			This subclass captures the characteristics of hypervisors that operate within a host OS for hosting guest operating systems inside of it.
Provider					This class captures the characteristics of IaaS or PaaS providers and extends the hierarchy and concepts used in the Saloon ontology (Quinton et al., 2012; Quinton et al., 2013)
				<i>offersIaaS</i>	This object property associates the Provider class with the IaaS

					class of the Application Placement model for describing its infrastructural offerings.
				<i>offersPaaS</i>	This object property associates the Provider class with the PaaS class of the Application Placement model for describing its platform level offerings.
				<i>hasProviderReputation</i>	This property refers to a string that denotes how appreciated is a certain provider based on its customers satisfaction.
				<i>hasGreenFootprint</i>	This property refers to a boolean value that denotes whether or not the provider's offerings have the minimum possible impact on the environment.
<b>PaaS</b>					This class encapsulates all the attributes related to platform level cloud resources that are required and offered for deploying Melodic-enabled applications.
				<i>isOfferedbyProvider</i>	This object property associates the PaaS class with the Provider Class (of the Application Placement Model) for expressing characteristics and identity of the resource provider.
				<i>usesCloud</i>	This object property associates the PaaS class with the Cloud Class (of the Application Placement Model) for denoting with one reference the characteristics of the underlying IaaS level resources used for offering PaaS services.

				<i>hasCloud Location</i>	This object property associates the PaaS class with the Cloud Location Class (of the Big Data Model) for expressing the network or physical location of the virtualised resource.
				<i>hasCost Function</i>	This property associates the PaaS class with a string that refers to a function that provides an accurate calculation of the expected cost for using platform level services.
				<i>has Availability</i>	This property may associate a PaaS resource with a float that expresses the expected uptime of the platform level resource.
				<i>hasPricing Type</i>	This Property refers to the different pricing per use schemes that each provider may offer regarding platform level cloud resources.
	Platform				This subclass is used to register and select any one of the different available PaaS offerings depending on the scope of their offered services (e.g. OpenShift, CloudFoundry etc.).
	Environment				This subclass encapsulates all the aspects that identify the platform level cloud environment (Höfer & Karagiannis , 2011).
				<i>hasOS</i>	This property describes the offered or requested operating system (e.g. UbuntuServer, CentOs). It associates the Environment class with a string.



		Framework			This subclass describes the offered or requested web framework for rapid development (e.g. PLAY, DJANGO)
		Image			This subclass describes any pre-installed cloud image available for initializing an IaaS resource.
				<i>hasImageId</i>	This property refers to an integer for referencing available cloud images.
				<i>hasLanguage Support</i>	This property refers to the offered or requested development language support which implies certain related middleware (e.g. Java runtime, .NET runtime etc.). It associates the Environment class with a string.
	Application Server				This subclass accumulates all the necessary application server that might be requested (e.g. Apache Tomcat 9.0.x, Jetty 9.3.3 etc.).
	PaaS Configuration				This subclass is used in order to register all the configuration details needed for using platform level cloud services.
				<i>hasVersion</i>	This property refers to a string for denoting the version of a certain configuration.
				<i>hasAPI</i>	This property refers to a string for denoting the API that can be used for performing application management (e.g. upload applications to the cloud, start/stop, monitor application etc.)

				<i>hasDownload</i>	This property refers to a string for denoting services needed to be downloaded and installed before hosting a cloud application.
--	--	--	--	--------------------	--

Table 3 is used for describing further details of the Application Placement model with respect to security controls offered or required as a service. This separate table is used for better readability since the Security Control class involves a bigger number of subclass levels than the rest of the top-level concepts of the Application Placement model.

Table 3: Security Controls Details

Class Taxonomy Levels		Properties	Description and Related Ontology (if any)
Security Controls			This is a subclass of the PaaS class and refers to all the possible security enforcement mechanisms that may be offered or required as a service for protecting the operation of hosted cloud applications. All its subclasses refer to specific security controls that have been classified based on the latest version of the Cloud Controls Matrix (CSA, 2016) introduced by the Cloud Security Alliance.
		<i>guarantees Non Repudiation</i>	This property refers to a boolean value that states whether or not the offered PaaS services can provide proof of the integrity and origin of data with high assurance.
	CSA-IAM-02		This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Identity &

				Access Management - Credential Lifecycle / Provision Management.
		Identity Management		This subclass is used to denote the offering or required functionalities for registering and updating the identity of entities that may request access to cloud resources or sensitive data along with their access requests and actions logging.
			Authentication	This is a subclass that encapsulates offered or required capabilities for attesting the identity of entities that may request access to cloud resources or sensitive data.
			Access Logging	This subclass refers to mechanisms and their characteristics offered or required for registering and persisting all kind of access actions performed.
			Credential Lifecycle Management	This subclass refers to end-user's credential creation, update, deletion or revocation that may be offered or required in certain access control paradigms (e.g. Role-based access control (RBAC)).
	CSA-IAM-09			This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Identity & Access Management - User Access Authorization.
		Authorization		This subclass is used to denote the offering or required

				functionalities for controlling the way access is permitted to cloud resources or persisted sensitive data (e.g. RBAC, Attribute-based access control).
	CSA-IVS-01			This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Infrastructure & Virtualization Security - Audit Logging / Intrusion Detection.
		IDS		This subclass is used to provide information about the characteristics of intrusion detection systems (IDS) offered or required for monitoring the virtual resource for malicious activities or any policy violations.
	CSA-IVS-06			This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Infrastructure & Virtualization Security - Network Security.
		IPS		This subclass is used to provide information about the characteristics of intrusion prevention systems (IPS) for examining network traffic flows and patterns in order to detect and prevent vulnerability exploits.
	CSA-IVS-13			This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Infrastructure & Virtualization

				Security - Network Architecture.
		DDoS Mitigation		This subclass is used to provide details on the distributed denial-of-service (DDoS) prevention capabilities offered or required for alleviating cyber-attacks that aim to constitute a cloud resource temporarily or indefinitely unavailable by flooding it with superfluous requests.
	CSA-GRM-10			This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Governance and Risk Management - Risk Assessments.
		Security Risk Assessment		This subclass lists the requested or offered tools for determining the security risks related to the virtualised resources use.
			CSA-IVS-05	This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Infrastructure & Virtualization Security - Vulnerability Management
			Vulnerability Assessment	This subclass of CSA-IVS-05 class is used to mention security vulnerability assessment tools offered or requested that accommodate the virtualization technologies used (i.e., virtualization aware) (CSA, 2016)

	CSA- EKM-02				This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Encryption & Key Management - Key Generation
		Key Management			This subclass is used to mention required or offered mechanisms necessary for creating, revoking and relaying cryptographic keys (to be used for encrypting/decrypting sensitive data) and also ensuring that these keys will not be revealed to any unauthorized or malicious users (Verginadis et al., 2015)
	CSA- EKM-03				This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Encryption & Key Management - Sensitive Data Protection
		Encryption			This subclass denotes the capability of offering encryption and decryption as a service from a certain virtualised resource.
	CSA- DSI-07				This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Data Security & Information Lifecycle Management - Secure Disposal
		Data Sanitization			This subclass denotes the capability of offering deliberate, permanent, and irreversible

					removal of data stored on a virtualised resource.
				<i>has Notification Method</i>	This property associates the Data Sanitization class with a string for requesting or stating the method used for informing the cloud resource's client about the successful sanitization of data.
	CSA-BCR-02				This class refers to all the relevant security controls offered as a PaaS service that belong to the CSA control domain entitled as: Business Continuity Management & Operational Resilience - Business Continuity Testing
		Security Testing			This subclass is used to mention any required or offered testing techniques and tools that verify the appropriate support of a certain virtualized resource for Confidentiality, Integrity, Authentication, Authorization, Availability and Non-repudiation. Example instances: FxCop <sup>3</sup> , FindBugs <sup>4</sup> , Appscan <sup>5</sup>

The details of the Application placement model are formally captured in the following UML Class diagrams where some of the most important data and object properties are revealed. For these diagrams, the Eclipse-based Papyrus<sup>6</sup> tool was used. Specifically, Figure 9 presents the details of IaaS and Provider classes, while Figure 10 presents the details of the PaaS classes (it is provided in two separate figures for better readability).

<sup>3</sup> <https://www.owasp.org/index.php/FxCop>

<sup>4</sup> <http://findbugs.sourceforge.net/>

<sup>5</sup> <http://www-03.ibm.com/software/products/en/appscan-source>

<sup>6</sup> <https://www.eclipse.org/papyrus/>

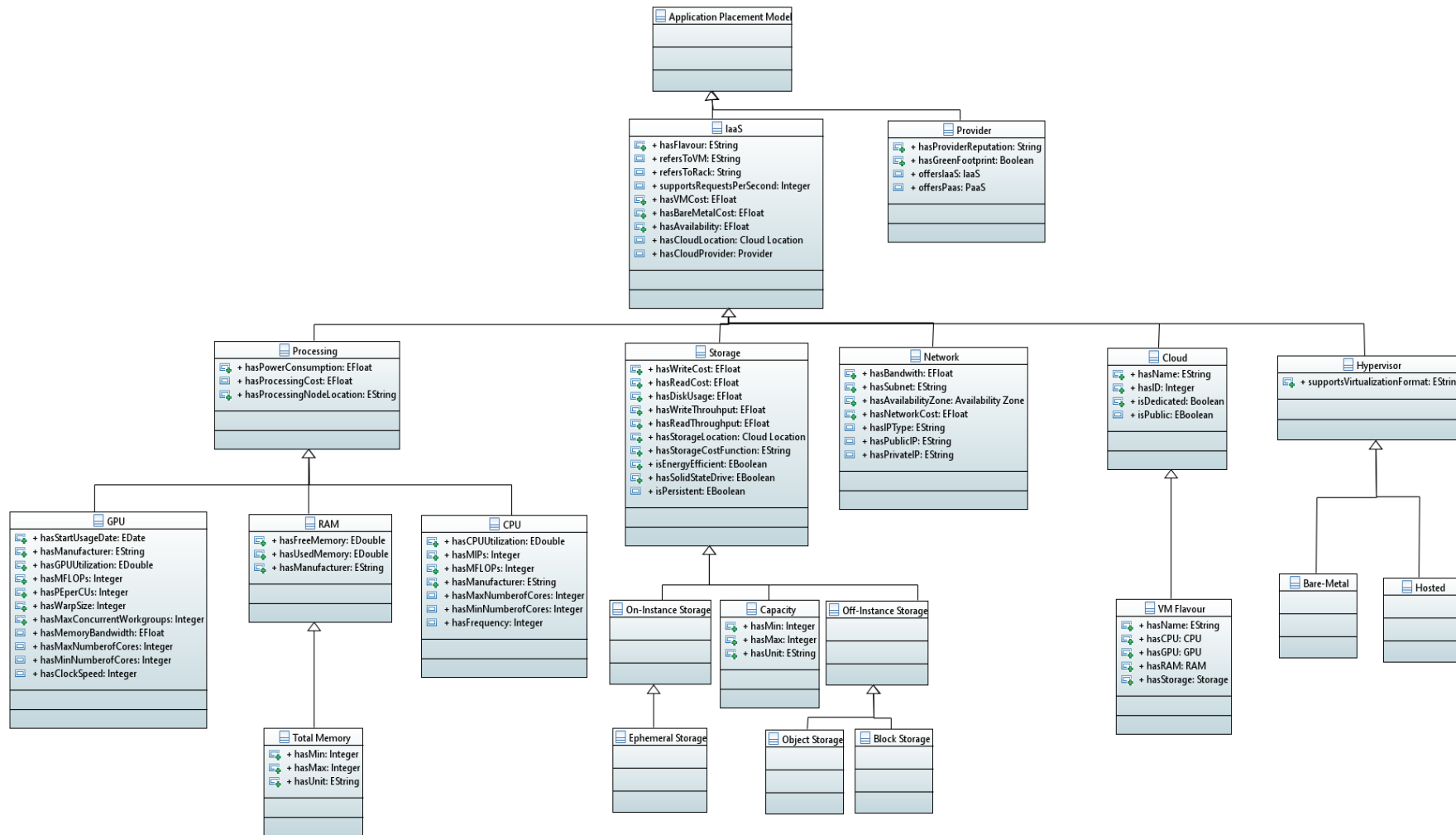


Figure 9: Application Placement Model's UML Class Diagram (1/2)



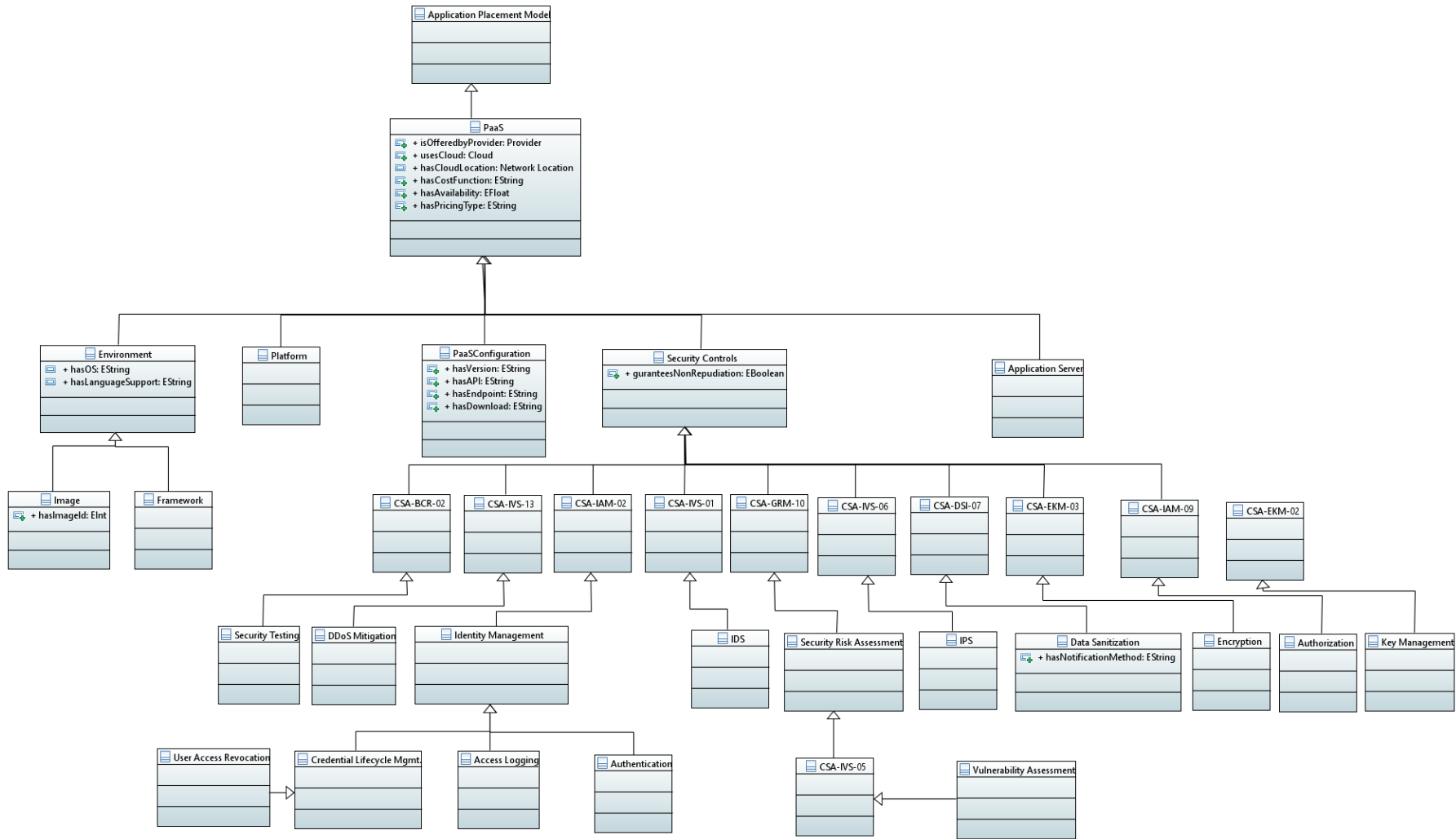


Figure 10: Application Placement Model's UML Class Diagram (2/2)

### 3.3 Big Data Model

#### 3.3.1 Big Data Model Overview

In the following Figure 11, Figure 12, and Figure 13, an overview of the core classes and sub-classes of the Big Data Model is provided. The overview diagram is provided in three separate figures in order to ensure its readability. Specifically, Figure 11 provides an overview of the Big Data Model depicting the details of its top level concepts in a two-level hierarchical tree. Figure 12 delves into further details (i.e. all the tree levels are revealed) for all the top-level concepts except from the Data Management class which is detailed in Figure 13.

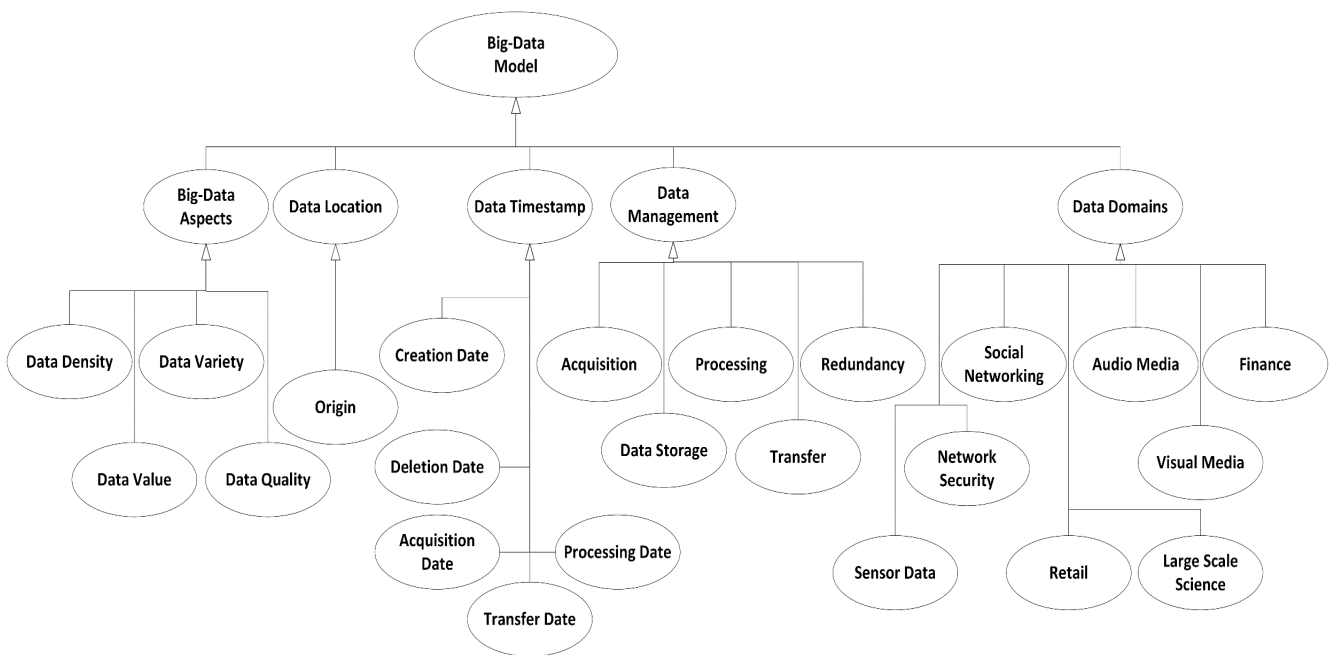


Figure 11: Big Data Model's Overview Diagram (1/3)

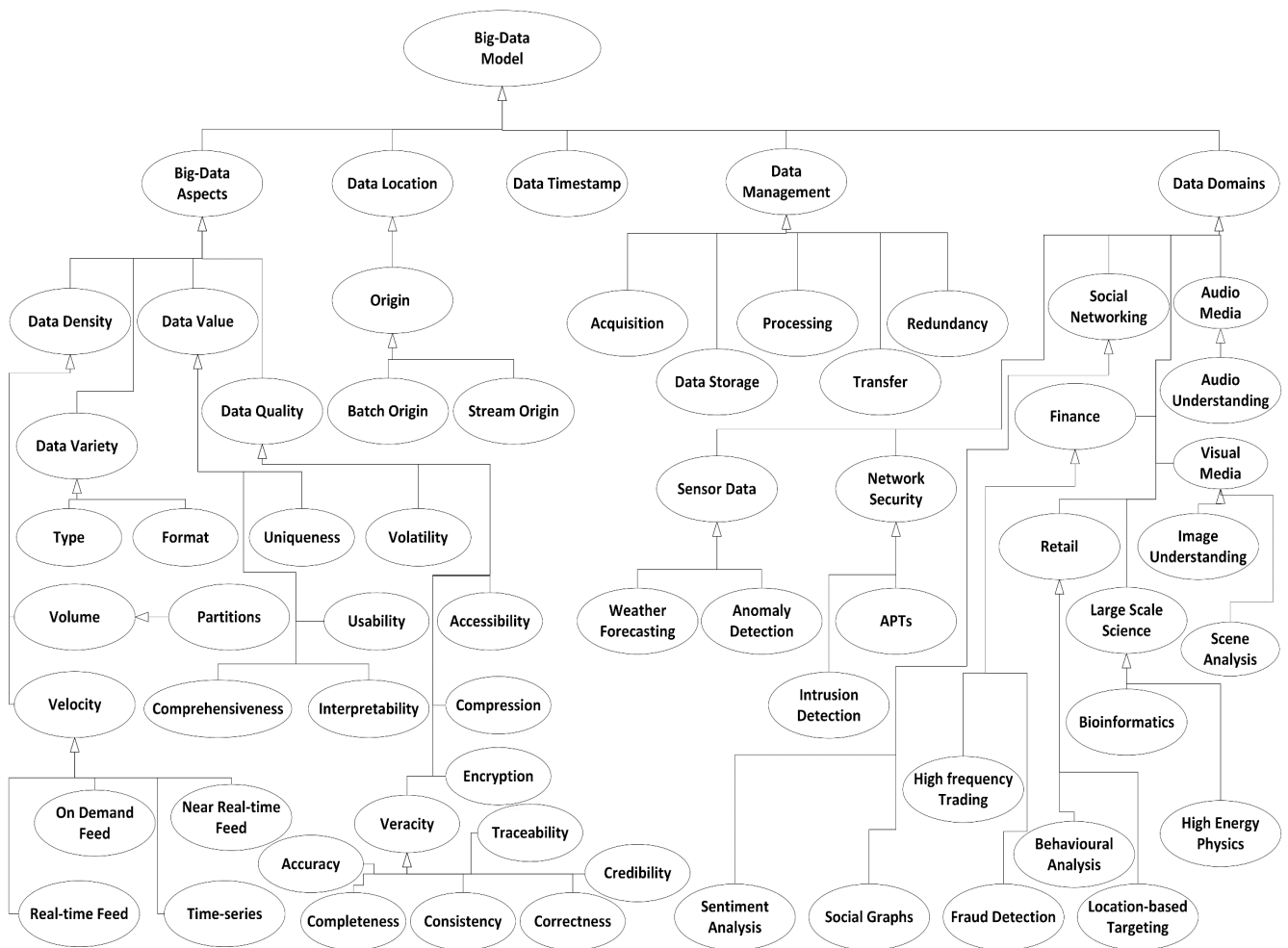


Figure 12: Big Data Model's Overview Diagram (2/3)

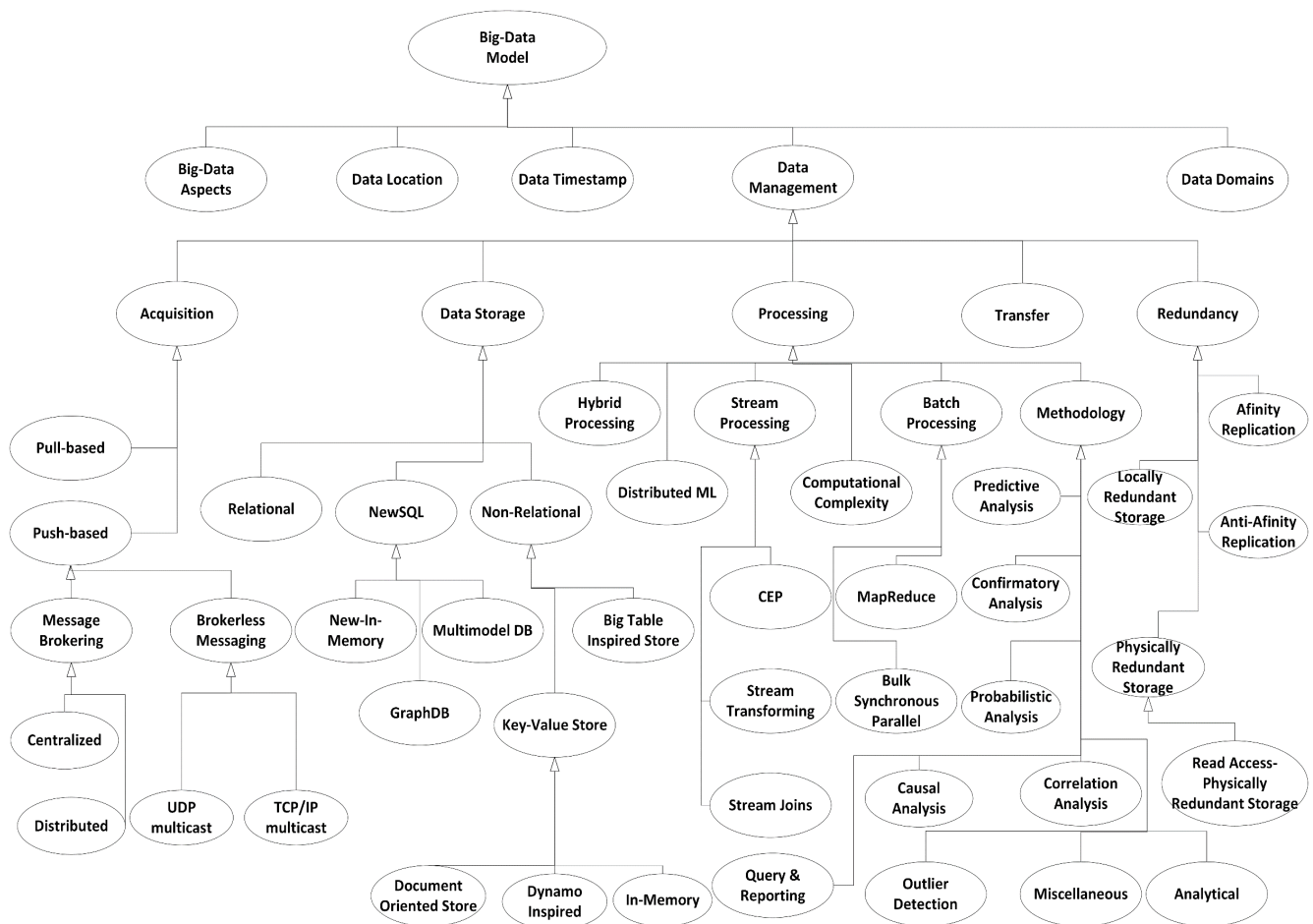


Figure 13. Big Data Model's Overview Diagram (3/3)

### 3.3.2 Big Data Model Details

The Big Data Model refers to the following top-level concepts.

- Big Data Aspects
- Data Location
- Data Timestamp
- Data Management
- Data Domains

For each of these top-level core classes, we provide their respective subclasses, their main properties along with their descriptions. This includes a detailed table where each concept is explained. Specifically, in Table 4, the details of the Big Data Aspects are presented while in Table 5, the Data Location and Timestamp concepts are discussed. In Table 6, we present Data

Management and in Table 7 the Data Domains details. We have used separate tables for discussing the Big Data Model in order to improve the tables' readability.

Table 4: Big Data Aspects Details

Class Taxonomy Levels				Properties	Description and Related Ontology (if any)
Big Data Aspects					This class encapsulates all the attributes that can be used in order to describe the main characteristics of big data to be processed by Melodic-enabled cloud applications hosted on multi-clouds. Based on such attributes, preferences on quantitative and qualitative dimensions of virtualized resources can be expressed.
				<i>hasData Owner</i>	This property associates the Big Data Aspects class with the Subject class of the Context Aware Security model in order to express the owner of the data to be handled by a Melodic-enabled application.
				<i>hasData Location</i>	This property associates the Big Data Aspects class with the Data Location class of the Big Data Model in order to denote where certain data artefacts may be found.
	Data Density				This subclass reveals details on big data observed or expected velocity and volume.
		Volume			This subclass reveals details on the expected amount of data artefacts to be processed by the Melodic-enabled cloud application.
				<i>canBe Partitioned</i>	This property associates the Volume class with a boolean value in order to clarify whether

					or not the data artefact can be physically fragmented into segments that are more easily maintained or accessed.
				<i>hasSize</i>	This property associates the Volume class with a float that describes the size of the data to be processed by the Melodic-enabled cloud applications (usually measured in gigabytes (GBs), terabytes (TBs), petabytes (PBs) etc.).
				<i>hasNumberOfRecords</i>	This property associates the Volume class with an integer that describes the size of the data expressed as the amount of records that it may involve.
				<i>fitsToMemory</i>	This property associates the Volume class with a boolean value in order to clarify whether or not the data artefact can fit in memory. We note here that we have included this property although by definition big data cannot fit in memory, in order to cover cases where there need to be a consideration of smaller amounts of data partitions in an overall big data intensive application.
			Partitions		This subclass is used to describe any details required for the data partitions to be used by Melodic-enabled cloud application.
		Velocity			This subclass reveals details on the anticipated speed of data to be processed by the Melodic-enabled cloud application along with the types of feeds that may be encountered (e.g. Real-time,

				On demand, Time-series, Continuous).
				<i>hasRate</i> This property is used in order to provide a measurement of how fast the data is produced, usually measure in megabits per second (Mbps) or gigabits per second (Gbps).
				<i>isInput Velocity</i> This property is used in order to measure the speed of data coming in to a Melodic-enabled application for processing (float value).
				<i>isOutput Velocity</i> This property is used in order to measure the speed of data produced by a Melodic-enabled application (float value).
				<i>is Continuous</i> This property refers to a boolean value that denotes whether data is produced and received in a continuous or in an intermittent way.
			Real-time Feed	This subclass implies aspects of the data velocity since it describes the access to or processing of data at the same time as it is produced. Such an access to data streams is possible in the order of milliseconds, and sometimes microseconds.
			Near Real-time Feed	This subclass describes that the access to or processing of data can be performed almost at the same time as it is produced. This implies a time delay introduced, by automated data processing or network transmission, between the occurrence of an event and the use of the data.

			On Demand Feed		This subclass describes that the access to or processing of data can be performed upon request. This implies that the data is produced either at the moment of the access request (e.g. get a sensor measurement) or it is first stored in order to be retrieved later on upon request.
			Time-Series		This subclass refers to data streams that can be characterized as series of data points indexed in time order.
	Data Variety				This class refers to the different types of data that should be processed by a Melodic-enabled cloud application, stating an increased diversity of data that should be stored, processed or combined.
		Format			This subclass refers to the structural variety that big data may involve which is expressed using certain schemes and models (e.g. binary large object (BLOB), JSON, XML etc.).
		Type			This subclass refers to the media variety that big data may involve with respect to the medium in which data get delivered (e.g. audio, image, video, text).
	Data Value				This class refers to big data aspects that reveal the business importance of data which is bound to the potential of improving a business entity's decision making capabilities.
		Uniqueness			This subclass refers to the amount of singular data sources involved and the level of their irreplaceability for defining



				aspects of the true value of the acquired data for a business entity.
		Usability		This subclass captures aspects of big data with respect to how relevant and useful is the data for a certain entity's business goals and decisions.
		Comprehensiveness		This subclass captures aspects of big data that are related to the semantic clarity, interpretability and thus usefulness of data acquired.
Data Quality				This class encapsulates another group of important big data concepts that reveal aspects about how accessible, secure, compact, volatile or uncertain the data is.
		Accessibility		This subclass refers to the level of convenience offered when attempting to access certain data artefacts. For example, accessing encrypted data might deteriorate the accessibility with the benefit of securing sensitive data.
		Volatility		This subclass refers to the degree of variation of data values over a period of time. The higher the volatility is the shorter the stored data can be considered valid.
		Compression		This class refers to if and how data has been encoded in order to use fewer bits than its original representation (i.e. as it was captured from the relevant data sources). This can be succeeded by identifying and eliminating statistical redundancy (lossless compression) or by removing

					unnecessary or less important information (lossy compression).
				<i>has Compression Algorithm</i>	This property associates the Compression class with string that mentions the compression algorithm used (e.g. Lempel-Ziv (1978))
				<i>hasBitRate Reduction</i>	This property associates the Compression class with string that denotes the compression ratio (e.g. 3,56 : 1 is the bit rate reduction when compressing 32 bytes to 9 bytes).
		Encryption			This subclass pertains to the details of the cryptographic paradigm used for protecting sensitive data by transforming a plaintext to ciphertext based on a cryptographic key.
				<i>has Encryption Type</i>	This property associates the Encryption class to a string that states the cryptographic algorithm used for performing the data encryption (e.g. Advance Encryption Standard (AES) (Daemen & Rijmen, 2003), RSA (Rivest et al., 1978)).
				<i>usesMessage Verification</i>	This property associates the Encryption class to a string that denotes the technique used for protecting the integrity and authenticity of a data artefact transmission. This may involve the use of a digital signature that is a mathematical scheme for verifying the real data source produced it (authentication), guarantee non-repudiation and certify that data was not altered in transit (integrity).

				<i>usesKeySize</i>	This property associates the Encryption class to an integer that corresponds to the length (in bits) of the cryptographic key used for encrypting or decrypting sensitive data.
				<i>usesBlockSize</i>	This property associates the Encryption class to an integer that corresponds to a fixed length string of bits upon which the cipher operates.
		Veracity			This subclass pertains to uncertainty due to data inconsistency, incompleteness and approximations that lead to reduced accuracy.
			Credibility		This subclass is used to describe the integrity of data acquired, which is bound to the credibility of the data sources and network communication used.
			Accuracy		This subclass is used to describe the exactness of data acquired, which implies lack of approximations and/ or lack of detailed measurements.
			Completeness		This subclass is used to describe the plenitude of data acquired, which implies the appropriate acquisition of all the relevant data needed at any given time.
			Consistency		This subclass is used to describe the lack of any corrupted or conflicting data that could appear due to error-prone backup processes or unreliable transfer mediums.
			Correctness		This subclass is used to describe the level of faultlessness of data acquired which implies that both the data sources and the network

communication used were validated as trustworthy.

Table 5: Data Location Details

Class Taxonomy Levels	Properties	Description and Related Ontology (if any)
Data Location		This class encapsulates all the concepts that can be used for describing the origin of data or the current or required physical/ network location where the data can be stored or processed by a Melodic-enabled application.
	<i>isStorage Location</i>	This data property associates the Data Location class with a boolean value that specifies whether or not the data location mentioned is where the data will be stored or processed.
	<i>sameAs</i>	This object property associates the Data location class to another Data location recursively in order to facilitate the expression of requirements that dictate the use of the same location(s) as the ones previously selected for other data artefacts.
	<i>notSameAs</i>	This object property associates the Data location class to another Data location recursively in order to facilitate the expression of requirements that forbid the use of the same location(s) as the ones previously selected for other data artefacts.
	<i>hasSparsity</i>	This data property associates the Data location class to a string that denotes how distributed (e.g. Low, Medium, High) are the data sources or data locations exploited for producing a dataset to be processed by a Melodic-enabled application.

			<i>hasPreferred Location</i>	This object property associates the Data location class to the Location class of the Context Aware Security Model in order to facilitate the expression of preferences for using certain network, physical and/or cloud location(s) in order to store or process data artefacts.
			<i>has Allowed Location</i>	This object property associates the Data location class to the Location class of the Context Aware Security Model in order to facilitate the expression of permitted network, physical and/or cloud location(s) for storing or processing data artefacts.
			<i>has Unacceptable Location</i>	This object property associates the Data location class to the Location class of the Context Aware Security Model in order to facilitate the expression of forbidden network, physical and/or cloud location(s) for storing or processing data artefacts.
			<i>hasPhysical Location</i>	This object property associates the Data Location class to the Physical Location from the Context Aware Security model in order to define the concrete physical region where data may be stored or processed.
			<i>hasNetwork Location</i>	This object property associates the Data Location class to the Network Location from the Context Aware Security model in order to define the network region where data may be stored or processed.
			<i>hasCloud Location</i>	This object property associates the Data Location class to the Cloud Location from the Context Aware Security model in order to define the relevant concepts with respect to the worldwide positioning of cloud offerings hosts.
	Origin			This subclass involves all the relevant concepts for defining the source location

					of the data artefacts to be processed by a Melodic-enabled application.
		Batch Origin			This subclass defines the source location of data artefacts to be stored and processed in batch mode by a Melodic-enabled application.
		Stream Origin			This subclass defines the source location of data artefacts to be stream processed by a Melodic-enabled application.
<b>Data Timestamp</b>					This class includes all the necessary concepts for describing the temporal characteristics of data artefacts to be processed by a Melodic-enabled application.
				<i>hasTimePoint</i>	This property associates the Data Timestamp class with the Instant class of the Context Aware Security model for referring to the precise point in time at which data was created, deleted, acquired, processed or transferred.
				<i>hasTimeInterval</i>	This property associates the Data Timestamp class with the DateTimeInterval class of the Context Aware Security model for referring to a period of time bounded by two time points, during which data was created, deleted, acquired, processed or transferred.
	Creation Date				This subclass is used for defining when a certain data element was created.
	Deletion Date				This subclass is used for defining when a certain data element was deleted.
	Acquisition Date				This subclass is used for defining when a certain data element was received by the Melodic-enabled application.
	Processing Date				This subclass is used for defining when a certain data element was processed.

	Transfer Date				This subclass is used for defining when a certain data element was transmitted from a data source.
--	---------------	--	--	--	--

Table 6: Data Management Details

Class Taxonomy Levels				Properties	Description and Related Ontology (if any)
Data Management					This class encapsulates all the relevant concepts that can be used in order to describe major technological choices with respect to how big data is acquired, stored, processed, transferred or replicated for redundancy reasons.
				hasData Timestamp	This object property associates the Data Management class with the Data Timestamp class of the Big Data model in order to express the time when certain data artefacts were acquired, processed or transferred.
				hasAgent	This object property associates the Data Management class with the Subject class of the Context Aware Security model in order to express the responsible entity for performing data acquisition, processing transferring and storage.
	Acquisition				This subclass is used in order to describe the required or offered types of big data acquisition in the frame of a Melodic-enabled cloud application devised to process it.
				isReliable	This data property associates the Acquisition class with a boolean value that captures

					whether or not the means of data acquisition required or offered guarantee the accuracy of the data received.
				<i>buffers Messages</i>	This data property associates the Acquisition class with a boolean value that expresses the capability of a Melodic-enabled application to resolve any bottlenecks by buffering the surplus data, in cases that the data acquisition rates are larger than the processing rates.
				<i>applies Backpressure</i>	This property associates the Acquisition class with a boolean value that denotes the capability of interrupting the data source transmission in cases that the receiver and its buffers are not able to receive additional data for a short period of time.
				<i>drops Messages</i>	This property associates the Acquisition class with a boolean value that refers to bottleneck situations being resolved by dropping any surplus data.
				<i>hasSource</i>	This property associates the Acquisition class with the Data Location class of the Big Data Model in order to describe the location of the data source to be used by a Melodic-enabled application.
		Pull-based			This subclass aims to capture concepts related to the pull-based paradigm for acquiring data, where there is a request for triggering the transmission of data which is initiated by the



					Melodic-enabled application, i.e. the receiver entity (Yang et al. 2017) and the receipt takes place in a synchronous manner.
		Push-based			This subclass aims to capture concepts related to the push-based paradigm for acquiring data asynchronously, where the request for a given transaction is initiated by the publisher (Yang et al. 2017).
				<i>isOrdered</i>	This property associates the Push-based class with a boolean value that states whether or not the certain push-based technique used for relaying big data, guarantees the time ordering of the received data before their processing takes place.
				<i>uses Acknowledgments</i>	This property associates the Push-based class with a boolean value that defines whether or not the data source will repeatedly attempt to re-send the data to the Melodic-enabled application until a receipt confirmation message is sent.
			Message Brokering		This subclass refers to a certain type of push-based acquisition of data where an intermediary software component undertakes the task of translating and routing data transparently to any given number of subscribed receivers (Melodic-enabled applications) that expect the acquisition of certain data in a pre-defined format (Hohpe & Woolf, 2004).

			Centralized		This is a subclass of the Message Brokering class where the push-based paradigm, for communicating data between producers and subscribers, is implemented with a central broker application, usually called enterprise service bus – ESB (Chappell, 2004). Examples include: WSO2 ESB <sup>7</sup> , jBoss ESB <sup>8</sup> , Mule ESB <sup>9</sup>
			Distributed		This is a subclass of the Message Brokering class where the push-based paradigm for communicating data uses several dispersed, but integrated software applications (also called distributed ESB) with message brokering capabilities, instead of just one centralised broker entity in order to avoid any performance bottlenecks (e.g. Apache Kafka <sup>10</sup> ).
			Brokerless Messaging		This subclass refers to a certain type of push-based acquisition of data where there are not intermediaries in the middle for translating and routing data, instead direct peer-to-peer communication between the data sources and the receivers (i.e. Melodic-enabled application) is considered for low latency and/or high transaction rate applications (ZeroMQ, 2008).

<sup>7</sup> <http://wso2.com/products/enterprise-service-bus/>

<sup>8</sup> <http://jbossesb.jboss.org/>

<sup>9</sup> <https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb>

<sup>10</sup> <https://kafka.apache.org/>

			UDP Multicast		This is a subclass of the Brokerless Messaging class and refers to the simultaneous group communication (multicast) using the User Datagram Protocol –UDP (Kurose & Ross, 2010). Examples include: StatsD <sup>11</sup> , Brubeck <sup>12</sup> .
			TCP/IP Multicast		This is a subclass of the Brokerless Messaging class and refers to the technique for one-to-many communication over an IP infrastructure in a network. (Kurose & Ross, 2010). Examples include: ZeroMQ <sup>13</sup> , MQTT <sup>14</sup> .
	Data Storage				This subclass encapsulates all the concepts that can be used for characterising the way that input or output data should be stored. The hierarchy involved updates the storage infrastructure taxonomy that (Murthy et al., 2014) presented.
		Relational			This subclass refers to databases used for persisting data that are structured in a way that capture and present relations between stored data artefacts (Codd, 1970). Examples include: MySQL <sup>15</sup> , PostgreSQL <sup>16</sup> .
		Non-Relational			This subclass refers to databases (also called NoSQL) used for persisting data that are not modelled using tabular

<sup>11</sup> <https://github.com/etsy/statsd>

<sup>12</sup> <https://githubengineering.com/brubeck/>

<sup>13</sup> <http://zeromq.org/>

<sup>14</sup> <http://mqtt.org/>

<sup>15</sup> <https://www.mysql.com/>

<sup>16</sup> <https://www.postgresql.org/>

				relations and present certain advantages over the relational databases, especially for big data since they offer design simplicity and more efficient horizontal scaling.
			Key-Value Store	This subclass refers to a non-relational data storage paradigm designed for storing, retrieving, and managing associative arrays based on keys. These associative arrays contain a collection of objects with many different fields within them (Tweed & James, 2010).
			<i>Dynamo-Inspired</i>	This is a subclass of the Key-Value store class that represents database systems that adopt a set of techniques that make extensive use of object versioning and application-assisted conflict resolution in order to form a highly available key-value storage system (DeCandia et al., 2007). In order to achieve this level of availability, Dynamo-inspired systems sacrifice consistency under certain failure scenarios. Examples include: Riak <sup>17</sup> , Voldemort <sup>18</sup> .
			<i>In-Memory</i>	This is a subclass of the Key-value Store class and refers to databases that rely on cloud resource's main memory for

<sup>17</sup> <https://github.com/basho/riak>

<sup>18</sup> <http://www.project-voldemort.com/voldemort/>

				persisting. Examples include: Memcached <sup>19</sup> , Aerospike <sup>20</sup> .
			Document Oriented Store	This is a subclass of the Key-value Store class that refers to a software application designed for performing CRUD operations over semi-structured data called document-oriented information. Examples include: MongoDB <sup>21</sup> , NosDB <sup>22</sup> .
			Big Table-inspired Store	This subclass involves compressed, high performance databases inspired by the Bigtable proprietary solution built by Google that devises a sparse, distributed multi-dimensional sorted map (Chang et al. 2008). Examples include: HDFS <sup>23</sup> , Cassandra <sup>24</sup> .
		NewSQL		This subclass refers to a type of parallel database management systems that provides the same scalable performance of non-relational systems while still maintaining the same level of transactional support (i.e. support the properties of Atomicity, Consistency, Isolation, and Durability – ACID) as the traditional relational databases (Murthy et al., 2014).

<sup>19</sup> <https://memcached.org/>  
<sup>20</sup> <http://www.aerospike.com/>  
<sup>21</sup> <https://www.mongodb.com/>  
<sup>22</sup> <https://www.npmjs.com/package/nosdb>  
<sup>23</sup> [https://hadoop.apache.org/docs/r1.2.1/hdfs\\_design.html](https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html)  
<sup>24</sup> <http://cassandra.apache.org/>

			New-In-Memory		This subclass refers to NewSQL databases that primarily rely on cloud resource's main memory for persisting data instead of employing a disk storage mechanism. Examples include: VoltDB <sup>25</sup> , H-Store <sup>26</sup> .
			GraphDB		This subclass refers to databases that use graph structures (with nodes, edges and properties) for storing and retrieving data. Some are implemented adopting the relational paradigm by storing the graph data in a table while others use a key-value store or document-oriented database for storage (Angles & Gutierrez, 2008). Examples include: Neo4j <sup>27</sup> , OnyxDB <sup>28</sup> .
			Multimodel DB		This subclass refers to database management systems that support multiple data models (e.g. document, graph, relational, and key-value models) in one integrated backend (Lu & Holubová, 2017). Examples include: ArangoDB <sup>29</sup> , OrientDB <sup>30</sup> .
	Processing				This subclass encapsulates all the concepts that can be used for describing and classifying the various types of big data processing that can be conducted by a Melodic-enabled cloud application. The

<sup>25</sup> <https://www.voltodb.com/>

<sup>26</sup> <http://hstore.cs.brown.edu/>

<sup>27</sup> <https://neo4j.com/>

<sup>28</sup> <https://www.onyxdevtools.com/>

<sup>29</sup> <https://www.arangodb.com/>

<sup>30</sup> <http://orientdb.com/orientdb/>

					<p>hierarchy introduced updates both the DICE model for big data intensive application (Gómez et al., 2016) and the computer infrastructure taxonomy presented by CSA Big Data Taxonomy (Murthy et al., 2014).</p>
				<i>hasRaw Metric</i>	<p>This property associates the Processing class with a string value that refers to a core conceptualisation of the CAMEL language (i.e. Raw Metric), which encapsulates raw measurements of specific non-functional attributes, produced by sensors. Several interesting raw metrics that could be used are: Latency, Service Time, Response Time, Hit Rate on Cache – number of times that data were retrieved from the cache.</p>
				<i>hasComplex Metric</i>	<p>This property associates the Processing class with a string value that refers to another core conceptualisation of the CAMEL language (i.e. Composite Metric), which encapsulates composite measurements of specific non-functional attributes, derived via mathematical formulas over other metrics (e.g. average CPU usage, mean response time, Ratio Reads to Writes).</p>
				<i>isLongLived</i>	<p>This property associates the Processing class with a boolean that expresses whether or not a certain processing job is chronic.</p>

				<i>hasData Processing Cost</i>	This property associates the Processing class with a float that expresses the cost for retrieving data to be processed by the Melodic-enabled cloud application.
				<i>has Processing Cost</i>	This property associates the Processing class with a float that expresses the cost for processing according to the designed functionalities of the Melodic-enabled cloud application.
				<i>hasApp Processing Location</i>	This property associates the Processing class with the Cloud Location Class (of the Big Data Model) for expressing the network or physical location where the application is or will be hosted.
				<i>has Constraints</i>	This property associates the Processing class with a string that denotes expressions over raw and complex metrics that bound the way processing will be performed.
				<i>hasPriority</i>	This property associates the Processing class with a string that denotes how urgently a certain processing job should be executed.
				<i>forProduction Usage</i>	This property associates the Processing class with a boolean that denotes whether or not the discussed processing refers to a production system.
				<i>isRealTime</i>	This property associates the Processing class with a boolean that denotes whether or not the processing takes place at the



					same time as input data is produced.
				<i>isNearReal Time</i>	This property associates the Processing class with a boolean that denotes whether or not the processing takes place almost at the same time as input data is produced. This property implies a time delay introduced due to network lag, between the data source and the processing location.
		Stream Processing			This subclass refers to a processing paradigm also known as event stream processing that transforms, in real or near-real time, an incoming stream of unbounded data records/events that are small, self-contained, immutable objects containing the details of somethings happened at some point in time (Kleppmann, 2017).
			CEP		This subclass refers to one type of stream processing that aims to identify patterns and analyse cause-and-effect relationships among streams of information (data) in real time, allowing for proactive or reactive effective actions in response to specific situations (Luckham, 2002). It usually involves digestion of data from multiple and heterogeneous data. Examples include: Esper <sup>31</sup> , SQLstream <sup>32</sup> .

<sup>31</sup> <http://www.espertech.com/esper/>

<sup>32</sup> <http://sqlstream.com/>

			Stream Transforming		This subclass refers to a second type of stream processing that (instead of detecting occurrence patterns) focuses on incrementally converting (e.g. sorting, grouping, aggregating) streams to an analysis-friendly schema based on a number of devised jobs. Examples include: Apache STORM <sup>33</sup> , Apache Flink <sup>34</sup> .
			Stream Joins		This subclass involves information about different pre-processing techniques for enhancing the input of a stream processing application (Kleppmann, 2017). Examples include: Stream-stream joins – combining two or more distinct streams, Stream-table joins – combining streams with data stored, Table-table joins – combining different types of stored data into a single stream.
		Batch Processing			This subclass refers to a processing paradigm also known as offline system that involves a series of jobs on a bounded bundle of inputs (fixed set of data) that produces a certain output (Kleppmann, 2017). Batch jobs are often scheduled to run periodically (e.g. once a day).
				<i>hasTotalTime PerJob</i>	This property associates the Batch Processing with a float that denotes the total time it lasts to execute a job on a

<sup>33</sup> <http://storm.apache.org/>

<sup>34</sup> <https://flink.apache.org/>

					dataset of a certain size (Kleppmann, 2017).
				<i>has Throughput</i>	This property associates the Batch Processing with an integer that states the number of data records that an application can process per second, or the total time it takes to run a job on a dataset of a certain size (Kleppmann, 2017).
			MapReduce		This is a subclass that refers to one of the most representative batch processing techniques for distributed computing; it takes a set of data and converts it into a new set of data (Map method – e.g. filtering, sorting) that it is then processed (Reduce method – e.g. summary operation), resulting in the braking down of individual elements into tuples (key/value pairs). Examples include: Hadoop MapReduce <sup>35</sup> , Apache S4 <sup>36</sup> .
			Bulk Synchronous Parallel		This subclass refers to a bridging model for designing parallel algorithms which fits the needs of distributed computation (Valiant, 2011). Examples include: Apache Hama <sup>37</sup> , Apache Giraph <sup>38</sup> .
		Hybrid Processing			This is a subclass that involves processing techniques that can be classified in the space between stream and batch processing. These are also

<sup>35</sup> <https://hadoop.apache.org/>

<sup>36</sup> <https://github.com/apache/incubator-s4>

<sup>37</sup> <http://hama.apache.org/>

<sup>38</sup> <http://giraph.apache.org/>

					called micro-batching techniques and treat streams as a sequence of small batches or chunks of data that are processed in near real-time (e.g. Apache Spark <sup>39</sup> ).
		Distributed ML			This subclass refers to big data processing techniques that are focused on implementing efficient machine learning algorithms in a distributed and scalable manner (Ranjan et al., 2015). Examples include: Apache Mahout <sup>40</sup> , MLBase <sup>41</sup> .
		Computational Complexity			This subclass is used in order to provide concepts that classify processing techniques and systems based on the computation complexity that they introduce when processing big data. Example instances may include the use of linguistic terms like High, Medium, Low or property values that imply the expected complexity (e.g. job length).
				<i>hasJobLength</i>	This property associates the Complexity class with an integer that denotes the size of each program used as a big data processing job.
				<i>hasRequired Processing Power</i>	This property associates the Complexity class with the CPU class of the Application Placement model in order to indirectly indicate the involved complexity of a certain Melodic-enabled application.

<sup>39</sup> <https://spark.apache.org/>

<sup>40</sup> <http://mahout.apache.org/>

<sup>41</sup> <http://www.mlbase.org/>

		Methodology			This subclass is used for characterising big data processing applications based on their scope and the kind of algorithmic approach used.
			Outlier Detection		This subclass refers to the use of anomaly detection algorithms for identifying items, events or observations which do not conform to an expected pattern or other items in a dataset (Chandola et al., 2009).
			Probabilistic Analysis		This subclass refers to the use of certain algorithms that based on big data processing may infer with a certain degree of certainty, facts about the current situation.
			Confirmatory Analysis		This subclass refers to the algorithms (often used in social research) corresponding to a special form of factor analysis that focus on testing and verifying hypothesized models (based on theory and/or previous analytic research) (Li, 2015).
			Predictive Analysis		This subclass refers to a number of techniques from data mining, statistics, predictive modelling, machine learning, and artificial intelligence employed to analyse big data in order to make predictions about future events (Nyce, 2007).
			Correlation Analysis		This subclass refers to statistical evaluation techniques used on big data processing to study the strength of a relationship

				between two, numerically measured, continuous variables (Rodgers & Nicewander, 1988).
			Causal Analysis	This subclass refers to techniques that introduce counterfactual reasoning and causal assumptions in addition to observations and statistical assumptions in order to estimate the effect of intervention between two variables (Pearl, 2003).
			Analytical	This subclass refers to the kind of big data processing that considers qualitative or quantitative information to discern patterns within the information, usually involving deductive reasoning. Examples include: Social Network Analysis, Text Analysis.
			Query & Reporting	This subclass refers to techniques and tools that perform simple or complex inquiries over a set of data and provide consolidated summaries of the results.
			Miscellaneous	This subclass refers to any other methodology used for big data processing that can be perceived as part of the above mentioned techniques. Examples include: 3D Reconstruction, Translation
	Transfer			This subclass of Data Management class refers to any concept that can be used for describing aspects related to communicating data artefacts between their data sources and

					the processing or storing locations.
				<i>hasData TransferCost</i>	This property associates the Transfer class with a float that denotes the actual or expected cost for transferring data.
				<i>hasData Transfer Duration</i>	This property associates the Transfer class with a float that denotes the time needed for transferring data between different locations.
				<i>hasTransfer Origin</i>	This property associates the Transfer class with the Data Location class of the Big Data Model in order to identify the source location of a data-transferring task.
				<i>hasTransfer Target</i>	This property associates the Transfer class with the Data Location class of the Big Data Model in order to identify the sink location of a data-transferring task.
				<i>hasData Transfer DesiredStart Time</i>	This property associates the Transfer class with a date datatype in order to define the desired start time of a data-transferring task.
				<i>hasData Transfer Desired Completion Time</i>	This property associates the Transfer class with a date datatype in order to define the desired end time of a data-transferring task.
	Redun- dancy				This subclass encapsulates any approach used for persisting the same data artefacts in several separate places, either in a single database, or in remote databases for detecting and reconstructing lost or

					damaged data (Doorn & River, 2002).
				<i>refersToVM Replication</i>	This property associates the Redundancy class with a boolean that mentions whether or not, for data redundancy reasons, an approach is used for taking a snapshot of a certain VM and copying it to another VM, situated either on the same or different physical hosts.
				<i>refersToPM Replication</i>	This property associates the Redundancy class with a boolean that mentions whether or not, a data redundancy approach is used for taking snapshots of a certain physical machine and copying it to another infrastructural resource.
				<i>has Synchronous Replication</i>	This property associates the Redundancy class with a boolean that defines whether or not data is written both to primary storage and the replica simultaneously, in order to accomplish a constant synchronization between data redundancy entities.
				<i>has Asynchronous Replication</i>	This property associates the Redundancy class with a boolean that defines whether or not data is written to the replica after it has been persisted to the primary storage.
				<i>hasCircular Replication Topology</i>	This property associates the Redundancy class with a boolean that defines whether or not a master slave topology is used for replicating data where each master (where new data initially is stored) is also the



					slave (that receives copies of the initial data) of another master, in a circular fashion.
				<i>hasStar Replication Topology</i>	This property associates the Redundancy class with a boolean that defines whether or not a central master node is used (where new data initially is stored) and a number of peripheral nodes are directly connected to the master node for replicating data.
		Affinity Replication			This subclass refers to replication techniques that consider or define resources to be used for replicating data.
		Anti-Affinity Replication			This subclass refers to replication techniques that consider or define resources to be excluded from replicating data.
		Locally Redundant Storage			This subclass refers to data redundancy that considers hosts for data replication situated on the same host.
		Physically Redundant Storage			This subclass refers to data redundancy that considers hosts for data replication situated in different physical locations.
			Read Access-Physically Redundant Storage		This refers to a kind of physically-redundant storage that is optimised only for data retrieval.

Table 7: Data Domains Details

Class Taxonomy Levels	Properties	Description and Related Ontology (if any)
-----------------------	------------	---

Data Domains					This class encapsulates all the relevant concepts that characterize data based on the industries that produce it or need to extract information from it (Murthy et al., 2014). Specifically, we reuse and extend the big data taxonomy introduced by the Cloud Security Alliance (Murthy et al., 2014).
	Sensor Data				This subclass refers to semi-structured data produced from hardware or software sensors in various volumes, velocity and veracity that may be used for predicting or reacting to situations.
		Weather Forecasting			This is a subclass of Sensor Data that refers to data carrying valuable meteorological measurement that can be used for predicting weather information in the future.
		Anomaly Detection			This is a subclass of Sensor Data that refers to data carrying valuable information for detecting irregularities on recognised long term trends and patterns (e.g. unexpected traffic congestion).
	Network Security				This subclass refers to unstructured or semi-structured data produced from hardware or software sensors in various volumes, velocity and veracity that may be used for recognising network security threats.
		Intrusion Detection			This is a subclass of the Network Security class that refers to data coming from sensors that may reveal unauthorised access to computer systems.
		APTs			This subclass stands for Advanced Persisted Threat (APT) and refers to data that can be used to detect continuous cyber-threats, in

				particular that of Internet-enabled espionage, using a variety of intelligence gathering techniques to access sensitive information (Dell SecureWorks, 2012).
	Social Networking			This subclass refers to structured or semi-structured data produced from software sensors in various volumes, velocity and veracity that may be used for providing meaningful insights on the status and important aspects of a social network (e.g. user acquisition data).
		Social Graphs		This subclass refers to data that reveal all the interconnections and relationships between the members of an online social network.
	Finance			This subclass refers to structured data produced from software sensors coming from different financial sectors that usually have significant velocity and may be used for imprinting and analysing real-time financial transactions.
		High Frequency Trading		This subclass of the Finance class involves data that imprint, represent and can be used to analyse financial trading actions of high velocity.
		Fraud Detection		This subclass refers to data relevant for exposing fraudulent financial activities.
	Retail			This subclass encapsulates aspects of structured data analysed usually in near real-time that focus on all the relevant aspects of retail-related transactions or facts that may affect them (e.g. tweets used for sentiment analysis).
		Behavioural Analysis		This subclass refers to data that may reveal patterns of selling or buying

					conducts that can be exploited in different retail sectors.
		Location-based Targeting			This subclass refers to location and preferences related data that can be used for targeted advertising actions.
	Large Scale Science				This subclass addresses all the different data artefacts that may be used or produced in large scale science experiments and activities. Such structured data are usually exploited through batch processing.
		Bio-informatics			This is a subclass of the Large Scale Science class that focuses on data related to bioinformatics.
		High Energy Physics			This is a subclass of the Large Scale Science class that focuses on data related to high energy physics.
	Visual Media				This subclass encapsulates aspects of unstructured data that are related to video analysis and may be processed in batch, near real-time or even real-time.
		Image Understanding			This is a subclass of the Visual Media class that addresses data related to detecting and comprehending images.
		Scene Analysis			This is a subclass of the Visual Media class that addresses data and video applications for analysing video scenes.
	Audio Media				This is an additional subclass with respect to the big data taxonomy introduced by CSA (Murthy et al., 2014) and refers to data related to audio applications (e.g. gun fire detection).
		Audio Understanding			This is a subclass of the Audio Media class that refers to analysis of data for speech recognition and interaction.

The details of the Big Data Model are formally captured in the following five UML Class diagrams where some of the most important data and object properties are revealed (diagram provided in separate figures for better readability). Specifically, Figure 14 presents the most important properties of the top-level concepts of the Big Data Model. Figure 15 delves into the details of big data Aspects, Figure 16 analyses the Data Location and Data Timestamp classes, while Figure 17 and Figure 18 provide insights on Data Management and Data Domains respectively.

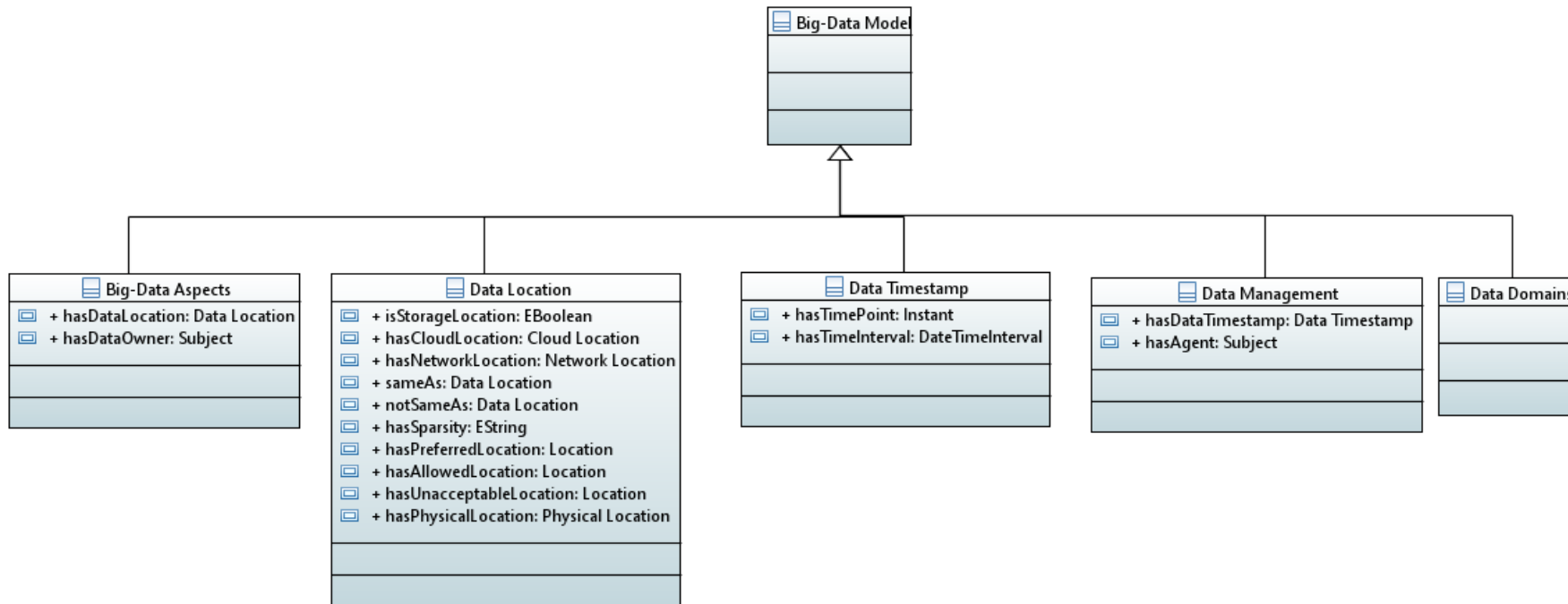


Figure 14: Big Data Model's UML Class Diagram (1/5)

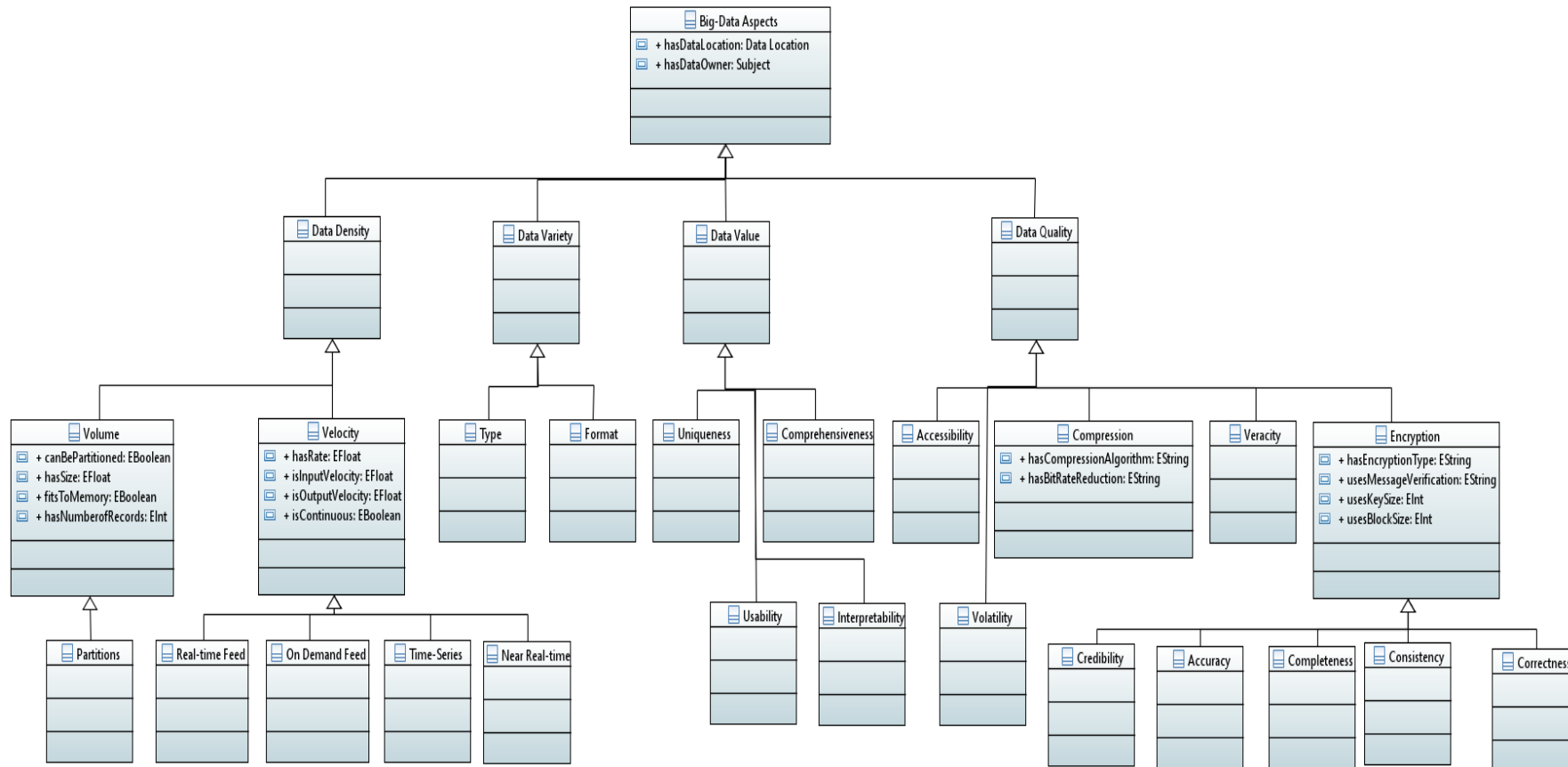


Figure 15: Big Data Model's UML Class Diagram (2/5)

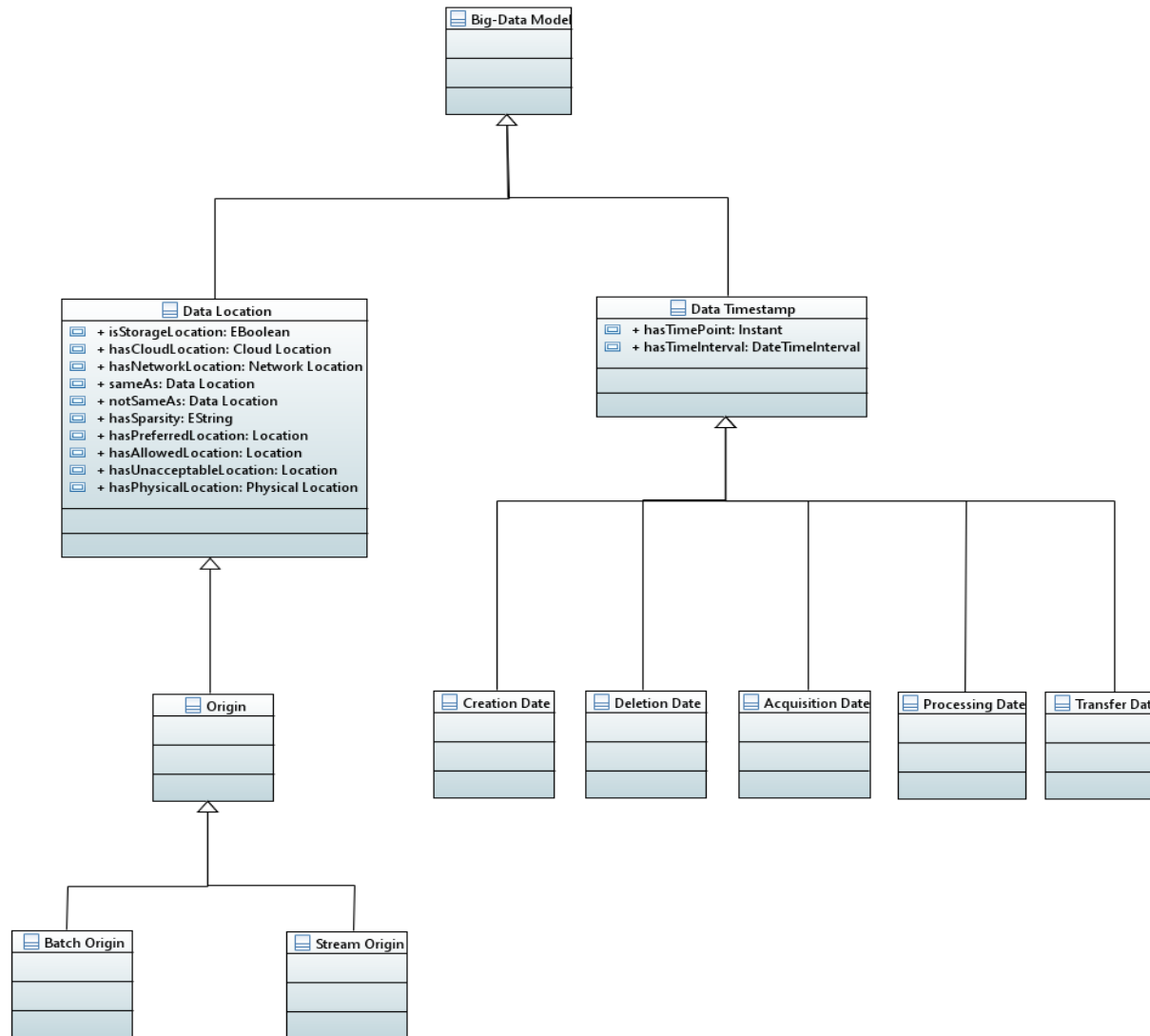


Figure 16: Big Data Model's UML Class Diagram (3/5)



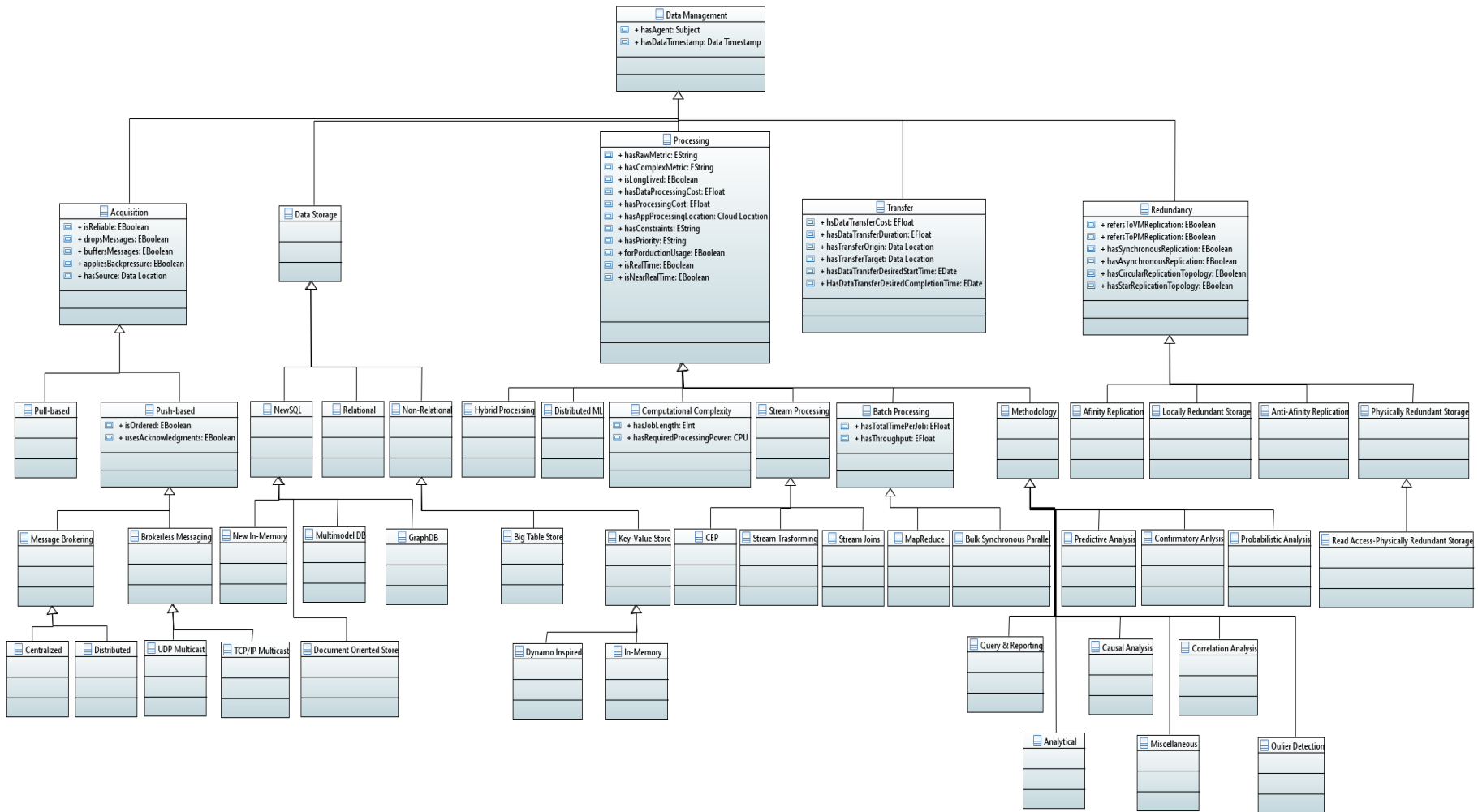


Figure 17: Big Data Model's UML Class Diagram (4/5)

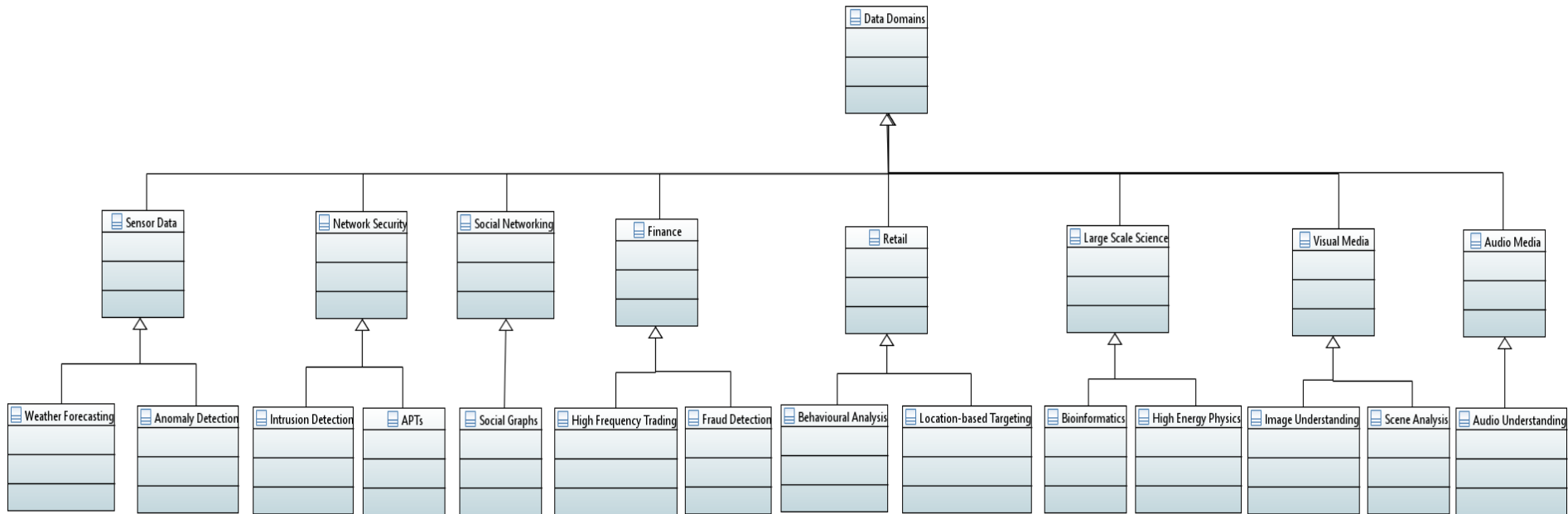


Figure 18: Big Data Model's UML Class Diagram (5/5)

### 3.4 Context Aware Security model

#### 3.4.1 Context Aware Security model Overview

As mentioned in Chapter 2, the third part of the Metadata Schema comes from the Context Aware Security Model that was developed in terms of the PaaSWord project (Verginadis et al., 2016; Veloudis et al., 2016). The overview of this model has already been presented in Figure 6. For the purposes of Melodic, we reuse this model and slightly extend it in order for it to serve as the background vocabulary of a PaaSWord-inspired context-aware access control engine. This engine will enhance with context-aware attribute-based access control capabilities (ABAC) certain Melodic components in order to protect the access, retrieval, storage or processing of any sensitive data artefacts. In Figure 6, the reader may find an overview of the Context Aware Security model as it has been described in PaaSWord project (Verginadis et al., 2016). We note that this slightly extended model will be incorporated as the third part of the Melodic Metadata Schema completing the Melodic vocabulary, while its semantic aspects will be exploited as part of the work on implementation, integration, and securing of Melodic components.

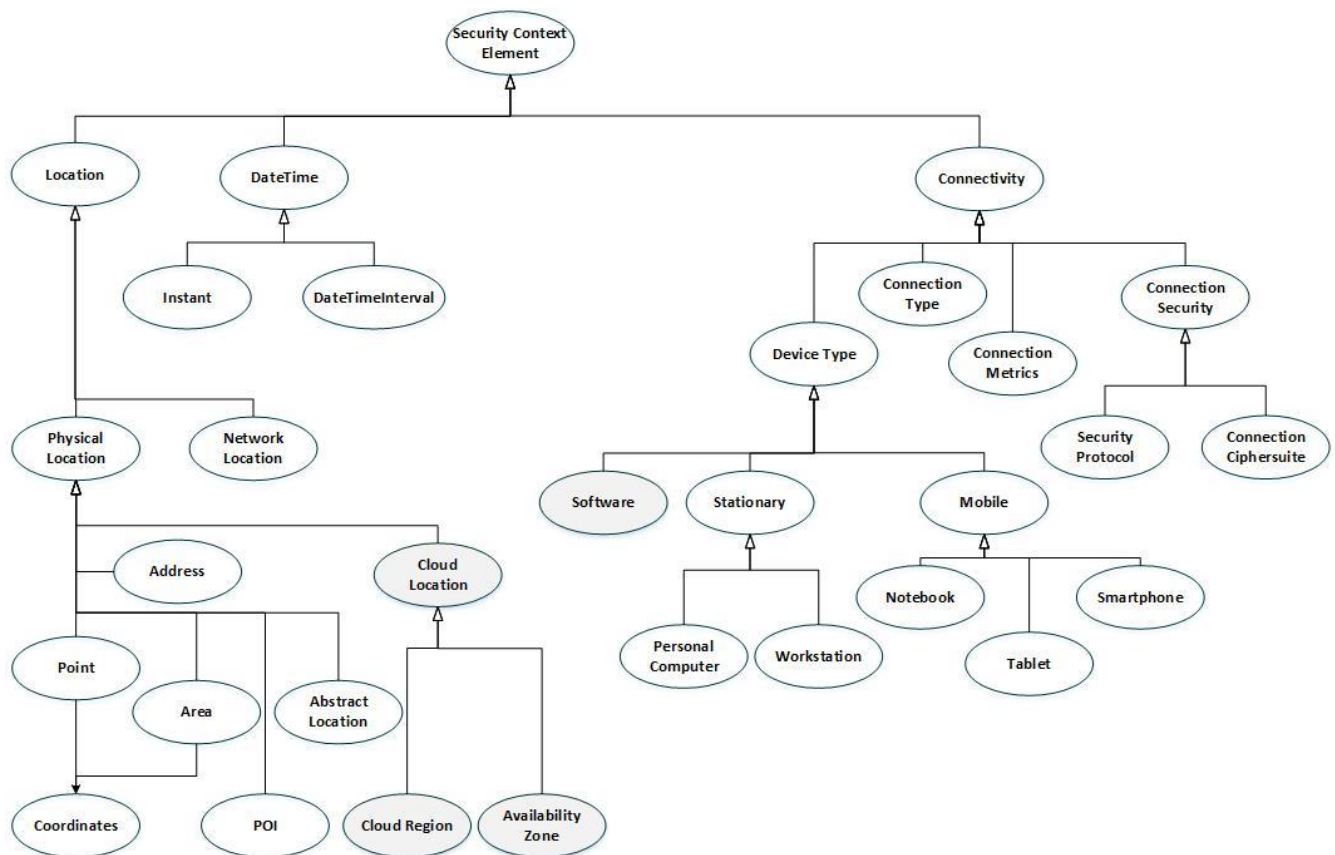
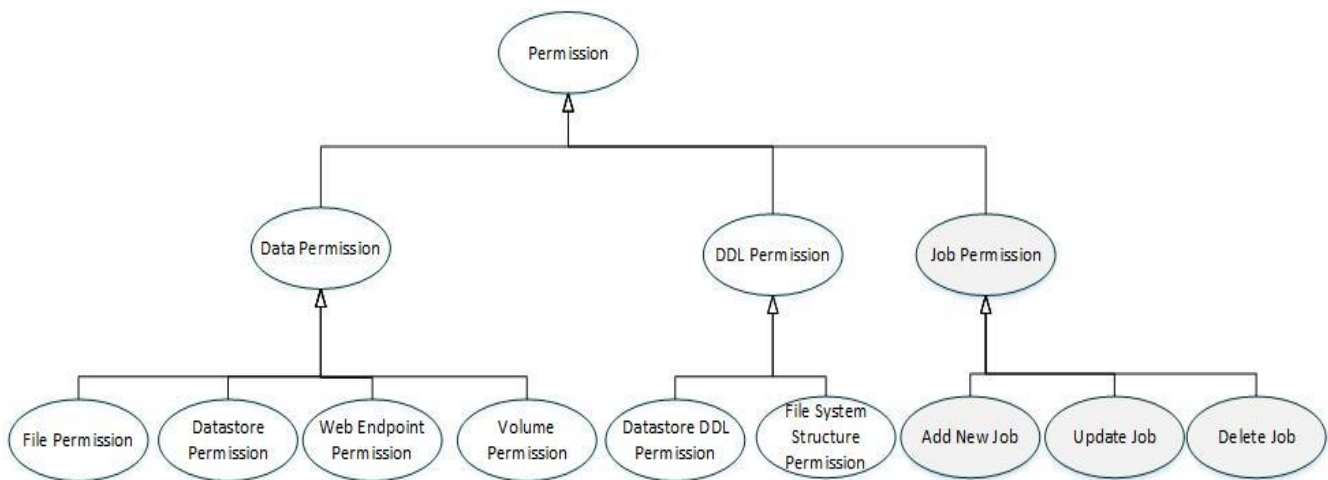


Figure 19: Security Context Element -Melodic Extensions

The following classes are included in the Context Aware Security Model that are briefly explained in Section 3.4.2:

- Security Context Element
- Permission
- Subject
- Object
- Context Patterns
- Handlers

Figure 19 and Figure 20 provide overview diagrams of the Security Context Element and Permission sub-models in order to highlight the basic extensions (depicted in grey colour) to this model from the Melodic perspective. These extensions are presented in detail in the following section 3.4.2.



*Figure 20: Permission - Melodic Extensions*

### 3.4.2 Context Aware Security model Details

This section includes a table that briefly explains the main classes of this model and provide details on the extensions proposed for Melodic purposes. Further details on the complete PaaSWord Context Aware Security model along with its formal description in UML class diagrams can be found in (Verginadis et al., 2016).

Table 8: Context Aware Security model Details

Class Taxonomy Levels				Properties	Description and Related Ontology (if any)
Subject					An instance of this class represents the agent seeking access to a particular data artefact. This can be an organization, a person, a group or a service (Verginadis et al., 2016)
	Software				This class represents software that attempts to access sensitive data (Verginadis et al., 2016)
		Melodic Component			(new class) This is a subclass that refers to any Melodic software mechanism that manages or decides where the data will reside, be transferred or be processed. This highlights that the Melodic authorization engine will mainly focus on access requests to sensitive data driven by Melodic components.
Object					This class refers to any kind of artefacts that should be protected based on their sensitivity levels. These artefacts may refer to relational and non-relational data, files, software artefacts that manage sensitive data, or even infrastructure artefacts used (Verginadis et al., 2016). Its subclasses include the Data Artefact, Software Artefact and Infrastructure Artefact.
Permission					(extended) This class refers to the actions that a Subject is allowed to perform upon an Object (Verginadis et al., 2016). Its subclasses as they were designed in PaaSword project are:  - Data Permission; This class refers to any action allowed by a Subject upon a data entity (e.g. datastore, file, web endpoint, volume access).

				- DDL Permission; This class reveals the data definition language (DDL) related actions on a specific Object (e.g. create/alter/drop datastore, Create/Delete/ChOwner a directory or file).
	Job Permission			(new class) This subclass refers to permissions related to the management of parallel computation tasks that get spawned into a number of resources, in response to a big data framework action (e.g. Spark).
		Add New Job		(new class) This subclass refers to permission concerning the introduction of a new big data processing job that should be implemented in terms of a big data framework.
		Update Job		(new class) This subclass refers to permission concerning the revision or enhancement of a big data processing job that has been implemented in terms of a big data framework.
		Delete Job		(new class) This subclass refers to permission concerning the exclusion of a big data processing job that has been implemented in terms of a big data framework.
<b>Security Context Element</b>				(extended) The security context element class refers to all the relevant classes and properties that capture Location, DateTime and Connectivity aspects, and characterize subjects, objects, requests and the environment related to an interaction (Verginadis et al., 2016). Below we provide details only for the Location and Connectivity classes that are extended.

	Location				This class describes a physical, a network and/or a cloud location where data are stored or from which a particular entity is requesting to access data (Verginadis et al., 2016).
				<i>Is Processing Location</i>	(new property) This property associates the Location class with a boolean value that denotes whether or not that certain location is referred as a position where data processing can/will take place (i.e. value equals to true) or as an area where data may reside (i.e. value equals to false).
		Network Location			An identifier for a node or network telecommunication interface from which a particular subject is requesting to access data or where the data resides (Verginadis et al., 2016).
		Physical Location			A physical location is a point or area of interest where data is stored, processed or from which a particular entity is requesting to access data. Physical locations might involve an address, a geographical position, an area, an abstract location and/or a Point of Interest [(Verginadis et al., 2016).
			Cloud Location		(new class) This subclass involves all the relevant concepts for positioning worldwide the hosts of cloud provider offerings. A cloud location is composed of Cloud Regions and Availability zones.
			Cloud region		(new class) This subclass of Cloud Location refers to separate geographical areas defined, where cloud provider datacentres may host virtualised infrastructure for storing or processing data (e.g. AWS – eu-west-1 located in Ireland).

			Availa- bility Zone		(new class) This subclass of Cloud Location refers to an isolated location out of multiple available ones per cloud region that defines an area where several cloud offerings may or should originate from (e.g. aws eu-west-1a, eu-west-1b, eu-west-1c). Usually, availability zones are interconnected with fast, private fibre-optic networking facilities.
	Con- nectivity				This class captures the information related to the connection used by the subject for accessing sensitive data (Verginadis et al., 2016). The following subclasses are included: Connection Type, Connection Metrics, Connection Security and Device Type
		Device Type			(extended) This class describes a device used when requesting access to sensitive data. Its subclasses include Mobile (i.e. portable device used when requesting access to sensitive data), Stationary (i.e. immobile devices used) and Software.
			Software		(new class) This third subclass of the Device Type refers to the use of middleware components for brokering the management of the access to sensitive data artefacts (e.g. Melodic's adapter that instructs the placement/storage of sensitive data on a certain virtualised resource).
				<i>isAuthen- ticated</i>	(new property) This property associates the Software class with a boolean value that denotes the digital sign of a certain software that may be allowed to manage the access to sensitive data. This indicates the adoption of a code-signing technology for guarantying the trustworthiness and authenticity of application used.



Request				<p>This class captures the characteristics that should be considered for evaluating an intercepted request (Veloudis et al., 2017)</p>
Context Pattern				<p>Context patterns are recurring motives of object accesses that are recognised in repeating context element instances. Future access requests on sensitive data can be decided also considering such information (Verginadis et al., 2016). Its subclasses are: Location pattern, DateTime pattern, Connectivity pattern, Object pattern, Permission pattern, Access Sequence Pattern.</p>
Handler				<p>This class refers to the characteristics of dedicated software components that are used for federating and processing raw data relevant to an access control decision and semantically uplifting them as instances of the Context Model (e.g. authentication, request, location, IP-address-to-city handlers etc. (Veloudis et al., 2017)</p>

## 4 CAMEL Updates based on Metadata Schema

CAMEL is a multi-DSL that covers multiple aspects in the description of multi-cloud applications. CAMEL is exploited in Melodic in order to provide support to all the phases of the multi-cloud application management. The current aspects covered include application structuring, requirements modelling, SLO & optimisation objective specification, security modelling, organisation modelling and execution history specification.

CAMEL is specified as an Eclipse Modeling Framework (EMF)<sup>42</sup> model based on the Eclipse<sup>43</sup> technology. Via such a model, concepts, their attributes and relationships between concepts can be specified. There is also a nice validation mechanism to further impose the semantics of the domain via the introduction of Object Constraint Language (OCL)<sup>44</sup> rules to complement existing restrictions on the meta-model level concerning, e.g., the specification of cardinality constraints on concept attributes or properties.

While CAMEL is quite rich and covers, as stated, multiple aspects, it still needs to go under a revision and further development process to cater the specific needs of Melodic. Each release cycle (e.g., 4-6 months) will end with a new CAMEL release. This release cycle has been set in order to enable a suitable amount of time to perform the respective code modifications or extensions in the platform in order to comply with new features or extensions of existing ones in CAMEL.

While the pursue of continuously improving CAMEL can enable it to become richer and more usable, it also comes with the cost of imposing additional effort requiring not only the update of the CAMEL meta-model, but also the retrieval of feedback which can support this enhancement. Such cost and effort is sometimes significant and further requires communicating and raising the level of understanding of the CAMEL evaluators so that the suitable feedback for change is received. This is due to the fact that the higher the level of understanding of the CAMEL meta-model is, the better and more suitable that feedback will be.

In order to reduce the updating effort, there is a need for a generic mechanism which allows CAMEL to evolve without actually affecting the meta-model level, which can become more or less stable over time. Such an evolution could be realised via the use of extensions at the model level, which are properly and sufficiently specified such that they can be processed by the Melodic platform in order to support the respective tasks (e.g., provider filtering) related to these extensions. The semantics of these extensions, however, would need to be made explicit such that the Melodic platform would be capable of properly interpreting them.

---

<sup>42</sup> <https://www.eclipse.org/modeling/emf/>

<sup>43</sup> <https://www.eclipse.org/>

<sup>44</sup> <https://wiki.eclipse.org/OCL>

In this respect, the Metadata Schema seems to be the most appropriate medium to realise this generic mechanism. First, it supplies a taxonomy of concepts which can be exploited for properly extending CAMEL with clear semantics, focusing mainly on specific aspects that can be extended (e.g., requirements, data aspect to be newly introduced). Second, it is editable and can be customised by the user according to his/her needs. In this respect, the user does not need to comply to a certain meta-model, but possibly make it closer to his/her requirements by providing the respective changes or extensions needed at a more abstract and independent (with respect to CAMEL or any other language) level. Third, it allows for introducing nice mechanisms for templating and instantiation. In particular, concrete instances of a certain concept can be immediately specified at the Metadata Schema level and then reused in the context of CAMEL specifications. For instance, specific data storage types could be specified and could be immediately used in the specification of data elements/objects in CAMEL. Last, it constitutes a nice playground for experimenting with possible changes and extensions of CAMEL which once being widely adopted can be exploited for further adapting CAMEL, if needed. For instance, the whole data aspect could be covered by the Metadata Schema; then through its usage in CAMEL based on the current mechanism, we could discern which are those elements that are widely used such that these are then incorporated into CAMEL.

While such a generic mechanism could seem easy to adopt, based on its initial conceptualisation, this is not the actual case in practice. It required a certain effort in order to modify CAMEL to make it suitable for incorporating this mechanism. In particular, different ways have been explored via which CAMEL could be extended using the Metadata Schema. The respective requirements, though, based on the current representation capabilities of the Metadata Schema, were quite clear: (a) enable the specification of new attributes in CAMEL for existing concepts; (b) enable the specification of new concepts at the model level as well as sub-concepts or attributes in them. Such capabilities lie on the fact that in the Metadata Schema, there are certainly all kinds of elements that can be found in a normal meta-model, like concepts, attributes and properties.

In the end, the integration realisation relied on the reuse of an existing internal DSL of CAMEL which is generic enough to specify any kind of concept or, better stated, *feature*. In particular, the provider meta-model of CAMEL was exploited based on its capability to specify tree-based structures of features along with the constraints that can be imposed on them. The metaphor to the meta-model level is obvious: features map to concepts, like *GPU*, while attributes map mainly to concept attributes, like *coreNumber*. Features can also internally include other sub-features, something that can simulate the properties/associations in a meta-model.

In this respect, the CAMEL extension follows a three-step approach. Initially, we desired to create a new, generic concept which could be used for connecting any element of CAMEL to a respective annotation in the Metadata Schema. To this end, the initial approach step involved the creation of the *NamedElement* concept, which represents a generic CAMEL element, including in its definition the following: (a) the name of the element; (b) the annotation of that element from the Metadata Schema (or any other semantic model that could be used to replace it); (c) a human-

understandable description of this element. As a second step, we have in parallel made *Feature* as a sub-concept of *NamedElement*, as well as most of the concepts in CAMEL as sub-concepts of *Feature*. In this way, a feature can really reference a Metadata Schema element while it can also comprise other sub-feature elements also mapping to this schema, thus enabling representation of feature trees. Moreover, as most CAMEL concepts are sub-concepts of *Feature*, they inherit the aforementioned characteristics and thus each one can be considered on its own as a feature tree. The final, third step involved the sub-classing of *Attribute* to *NamedElement*. Via this step, the following can be achieved: (a) we correlate an attribute with a respective element/attribute in the Metadata Schema; (b) we enable all kinds of features, thus almost all CAMEL elements, to include arbitrary attributes which do correlate with the attributes of the Metadata Schema. As a final result of this three-step extension approach, almost all CAMEL elements can be arbitrary enriched with whole feature models which can be associated with external ontologies or Metadata Schemata. The corresponding extension of CAMEL is depicted in Figure 21.

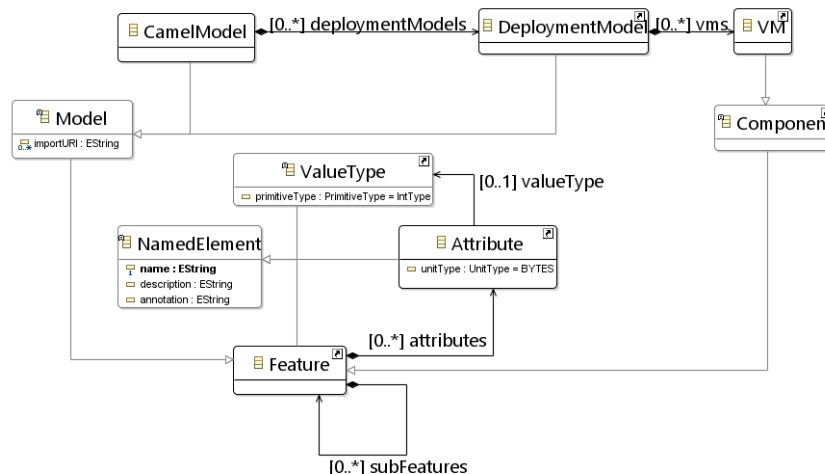


Figure 21: A snapshot of CAMEL focusing on its new concepts and extensions

In order to better explain this mechanism, we provide an example of how a CAMEL model can be enriched with additional content that is drawn from the Metadata Schema. Suppose that we need to additionally specify for a certain *VM* (type) that it needs to include a GPU for which the number of cores should be greater or equal to 2. In this respect, as a *VM* is now a *Feature*, we can create a new sub-feature for it, which is named as *GPU* and maps to the respective (equivalently named) concept in the Metadata Schema. For this sub-feature, we will create an attribute which will be named as *coreNumber* and which will be also associated with the *NumberOfCores* attribute in the Metadata Schema. Such an attribute will then have a certain value type which would map to the required range of values (i.e., the integer set  $[2, +\infty)$ ). A similar modelling is symmetrically performed at the provider model side. In particular, a cloud provider model would, naturally in our

case as we are using the provider/feature meta-model, specify a *VM* feature which would then include a *GPU* feature. It would also include an inter-feature constraint which will indicate which VM flavour (enumerated value of the *vmType* attribute of the *VM* feature) maps to which number of cores for its internal GPU via an inter-attribute constraint (involving the *vmType* and *coreNumber* attributes). Suppose that there is a VM flavour named "Medium" for which the number of cores of its GPU unit is 3. Then, the Profiler in the Melodic platform, would take both the CAMEL model and the provider one, and perform the matchmaking by relying on the fact that it needs to take into consideration the annotation of attributes and their type. In this way, if we are dealing with an attribute that maps to the same annotation from the Metadata Schema which is numeric in nature, then the matchmaking would succeed if the constraint over the required attribute capability should be less restrictive than the one of the provided attribute capability. In our case, this actually holds (as the value of 3 is included in  $[2, +\infty)$ ) which would then lead to a potential match between the VM requirement and respective VM capability/flavour offered. Figure 22 depicts a combined CAMEL model which includes the specification of this VM requirement and capability.

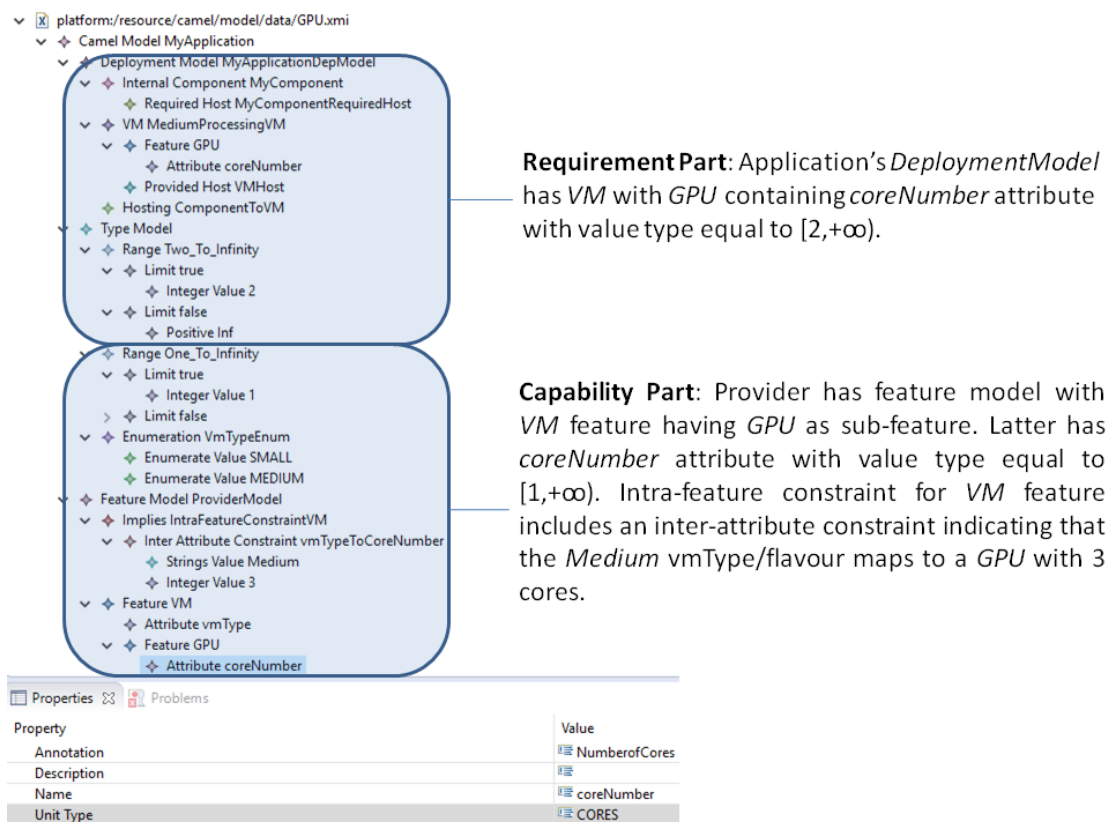


Figure 22: CAMEL snippet showing how GPU capabilities and requirements can be specified

We will now attempt to explain the templating capabilities that are additionally enabled via the integration of the metadata schema with CAMEL. This will be performed via two main examples/cases. The first example explains the forward direction in templating. In particular, suppose that in the Metadata Schema we have specified specific locations of a certain cloud which are specified in the form of a hierarchy. Such a specification, which could be done via the Metadata Schema editor, could then be easily transformed into a CAMEL location model. Such a location model would then be used for expressing location requirements in CAMEL application models, thus providing a very good medium of reuse of specific CAMEL elements. A depiction of how this could be performed from the Metadata Schema to CAMEL is supplied in Figure 23.

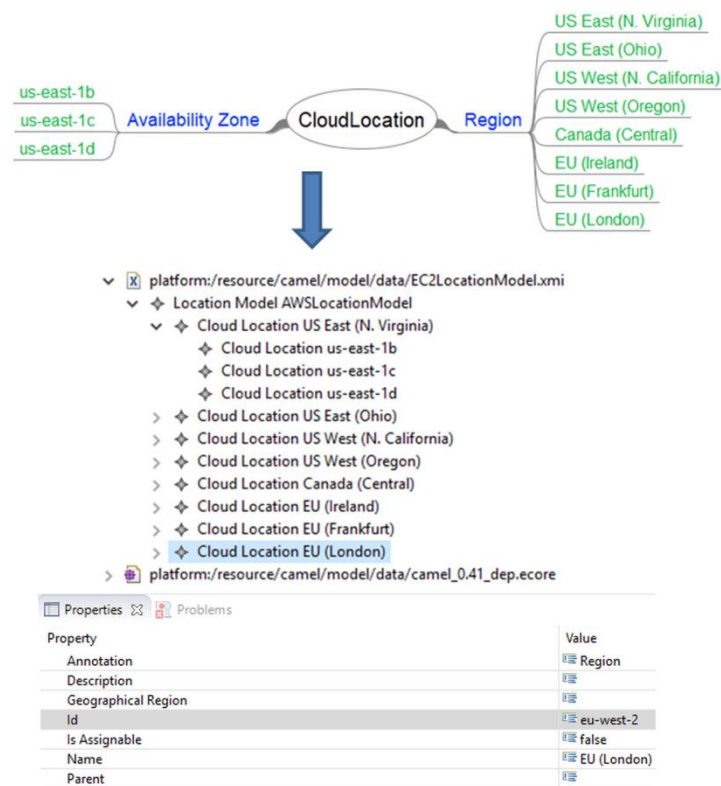


Figure 23: The transformation from Metadata Schema to CAMEL for Amazon AWS cloud locations

The second example concerns the templating of metrics. In this case, the templating capability is bi-directional. In particular, the user can start with a metric hierarchy specified in the Metadata Schema. Such a hierarchy could then be browsed by the CAMEL modeller in order to produce the respective full specification of some selected metrics in CAMEL, resulting in a template metric model which can be reused for the specification of SLOs or optimisation requirements in CAMEL requirement models. Then the concretisation of some metrics in the hierarchy could be forwarded at the Metadata Schema level, which could then depict via the editor which metrics have been already realised in CAMEL and which are not. This enables the user to have an account over which metrics can be used immediately and which ones need first to be specified in CAMEL before they can be exploited. Such a bi-directional mechanism also indicates a cooperation between the



meta-schema and CAMEL editors towards the specification of concrete CAMEL models from concepts of the Metadata Schema, which will offer a unification point towards the production of a uniform UI component in the Melodic platform. Such a cooperation could really provide good support to the user towards transforming his/her initial ideas/conceptualisations into concrete or template models in CAMEL for further reuse.

The concept of bi-directional templating is graphically depicted in Figure 24 where we can see an indication in a hierarchy of metrics about which metrics have been mapped to CAMEL realisations, and then we depict a CAMEL excerpt in which such a realisation is shown.

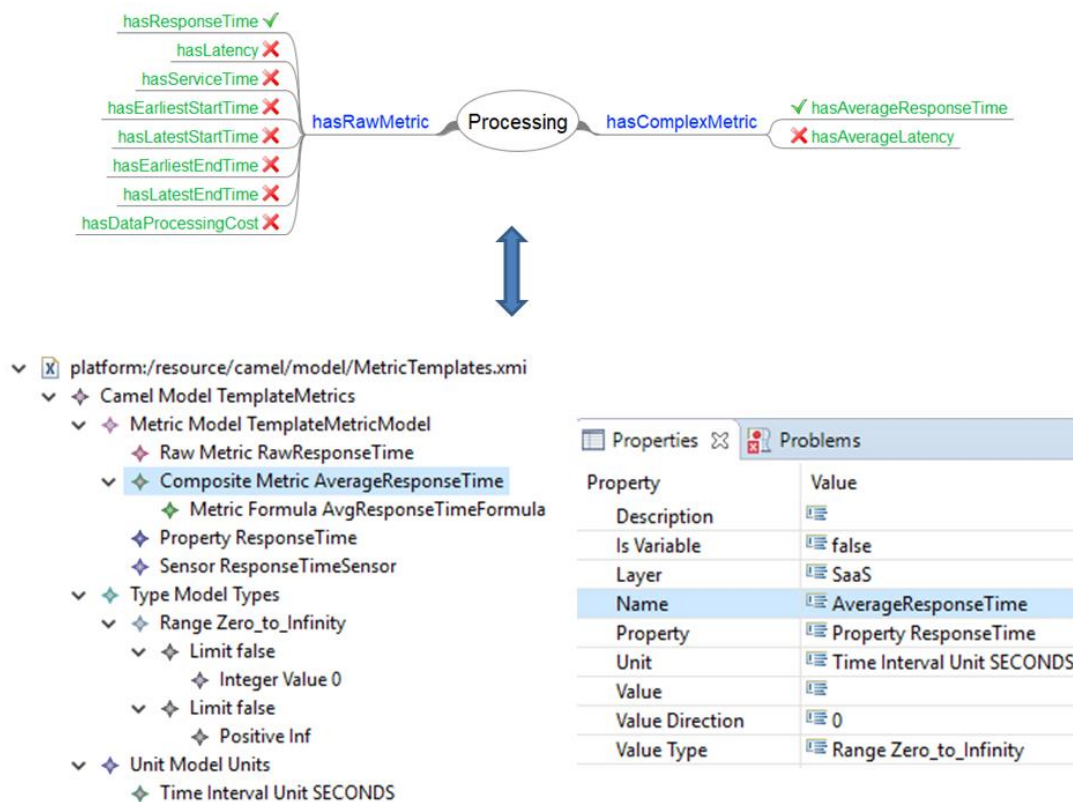


Figure 24: The bidirectional templating for metrics

## 5 Conclusions

In this deliverable, we introduced and discussed the details of the initial design of Melodic's Metadata Schema for data-aware multi-cloud computing. It corresponds to a vocabulary based on which the Melodic components are able to interpret requirements, constraints and offerings' characteristics in order to properly manage big data, optimise the placement of their processing jobs and control all accessing requests in multi-cloud environments. The Schema comprises the *Application Placement*, *Big Data* and *Context Aware Security* models that include classes and properties for defining where a certain big data application should be placed, what are the unique characteristics of the data artefacts that it needs to process, and what are the contextual aspects that may be used for bounding the access to the sensitive data.

Furthermore, we discussed the envisioned approach for extending critical aspects of the CAMEL language based on this Melodic vocabulary. We expect that these extensions or additions will seamlessly affect the Requirement, Metric, Scalability, Location, Provider and Security sub-models of CAMEL. Certain aspects of the Metadata Schema will be automatically weaved in CAMEL language based on work conducted with regards to the Melodic Upperware, that among others, involves an extended CAMEL and a Metadata Schema editor. Specifically, the Schema is extensible, and any Melodic adopter will be allowed to amend it according to their organisation's needs. We are currently developing a dedicated editor that on one side will enable CRUD operations over this Schema, and on the other side, will provide the necessary functionalities for retrieving cloud application developers' or DevOps' preferences over a number of qualitative criteria based on the Metadata Schema.

Concluding this report, we note that the data structures, of which instances describe particular cloud platforms, users, organisations, applications and applications' reconfigurations, may be relevant to be represented in a catalogue available to the Melodic Upperware and Executionware. We will examine the value of such a catalogue that may exploit the vocabulary defined in this work and that will be able to store the results of deployment and execution actions to provide the factual basis for improvement of the deployment choices. Thus, we may consider a single conceptual catalogue, utilising the conceptual level structures described in this deliverable. Such catalogue will have to represent the WoI (World of Interest) of Melodic so that the processing in the Upperware and Executionware achieves the real-world (business) objectives. To represent accurately the ontological structures the catalogue has to be encoded with formal syntax (for reliable processing) and declared semantics (to ensure the meaning of terms is understood). Furthermore, since legacy information sources of relevance may utilise a variety of metadata formats, there will be a need for conversion between them. If there are  $n$  such metadata schemes and we interconvert everyone to every other, we have a '*n-squared*' (formally  $n*(n-1)$ ) problem. If we convert all to a single canonical model the problem reduces to a scale of  $n$ . Increasingly, EC-funded ESFRI (Research Infrastructure) projects are utilising CERIF (Common European Research



Information Format – itself an EU Recommendation to Member States)<sup>45</sup> as the canonical model. It has conversion to/from DC (Dublin Core) and DCAT (Data Catalogue Vocabulary) both of which are W3C (World Wide Consortium) recommendations; CKAN (Comprehensive Knowledge Access Network used widely in open government data), ISO19115/INSPIRE (the latter an EU directive for geolocation metadata) and the ISO19139 XML representation and native RDF (Resource Description Framework, another W3C Recommendation). In Melodic, we will consider the use of CERIF for the Melodic catalogue.

---

<sup>45</sup> <http://www.eurocris.org/cerif/main-features-cerif>

## References

- Angles, R. Gutierrez, C., (2008). Survey of graph database models. *ACM Computing Surveys*. Association for Computing Machinery, 40 (1).
- Aslett, M., (2011). How Will The Database Incumbents Respond To NoSQL And NewSQL?. 451 Group. Available online at: <http://cs.brown.edu/courses/cs227/archives/2012/papers/newsq/aslett-newsq.pdf>
- Benavides, D., Segura, S., Cortés, A., R., (2010). Automated analysis of feature models 20 years later: A literature review. In: *Inf. Syst.* 35.6 (2010), pp. 615–636. doi: 10.1016/j.is.2010.01.001.
- Bernardi, S., Merseguer, J., Petriu, D., (2013). *Model-driven Dependability Assessment of Software Systems*. Springer.
- Chandola, V., Banerjee, A., Kumar, V., (2009). Anomaly detection: A survey. *ACM Computing Surveys*. 41 (3): 1–58. doi:10.1145/1541880.1541882.
- Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., Gruber, R. E., (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2), 4.
- Chappell, D., 2004. *Enterprise Service Bus*. O'Reilly, ISBN 0-596-00675-6.
- Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Olukotun, K., & Ng, A. Y. (2007). Map-reduce for machine learning on multicore. In *Advances in neural information processing systems* (pp. 281-288).
- Codd, E.F., (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*. 13 (6): 377–387. doi:10.1145/362384.362685.
- CSA, (2016). Cloud Controls Matrix. Available online at: [https://cloudsecurityalliance.org/group/cloud-controls-matrix/#\\_overview](https://cloudsecurityalliance.org/group/cloud-controls-matrix/#_overview)
- Daemen, J., Rijmen, V., (2003). AES Proposal: Rijndael. National Institute of Standards and Technology. Available online at: <https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1>
- DeCandia, G., Hastorun, D. Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Voshall, P., Vogels, W., (2007). Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP '07)*. Stevenson, Washington, USA: ACM. pp. 205–220. ISBN 978-1-59593-591-5.
- Dell SecureWorks, (2012). Anatomy of an Advanced Persistent Threat (APT). Available online at: <https://www.secureworks.com/resources/sb-advanced-threat-protection>
- Doorn, J. H., Rivero, L., C., (2002). Database integrity: challenges and solutions. Idea Group Inc (IGI). pp. 4–5. ISBN 978-1-930708-38-9.

- Frankel, D., S., (2003). Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons, ISBN 0-471-31920-1
- Gómez, A., Merseguer, J., Di Nitto, E., Tamburri, D., A., (2016). Towards a uml profile for data intensive applications. In Proceedings of QUDOS'16, pages 18–23, New York, NY, USA, 2016. ACM.
- Gruber, T., R., (1993). A translation approach to portable ontology specifications. In: Knowledge Acquisition 5.2, pp. 199–220. issn: 1042-8143. doi: 10.1006/knac.1993.1008.
- Höfer, C., N., Karagiannis, G., (2011). Cloud computing services: taxonomy and comparison. J Internet Serv Appl, 2:81–94, DOI 10.1007/s13174-011-0027-x
- Hohpe, G., Woolf, B., (2004). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions . Addison-Wesley, ISBN 0321200683.
- Kang J., Sim, K., M., (2011). Ontology and search engine for cloud computing system. International Conference on System Science and Engineering (ICSSE), pp. 276-281.
- Kleppmann M., (2017). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, ISBN:978-1-4919-0308-7
- Kritikos K., Massonet, P., (2016). An Integrated Meta-Model for Cloud Application Security Modelling. In Cloud Forward, 97 ( 2016 ) 84 – 93
- Kritikos, K., Domaschka, J., Rossini, A., (2014). SRL: A scalability rule language for multi-cloud environments. In IEEE International Conference on Cloud Computing Technology and Science, Singapore.
- Kurose, J. F., Ross, K. W., (2010). Computer Networking: A Top-Down Approach (5th ed.). Boston, MA: Pearson Education. ISBN 978-0-13-136548-3.
- Li, C.-H. (2015). Confirmatory factor analysis with ordinal data: Comparing robust maximum likelihood and diagonally weighted least squares. Behavior Research Methods. 48 (3): 936–949., doi:10.3758/s13428-015-0619-7.
- Lu, J., Holubová, I., (2017). Multi-model Data Management: What's New and What's Next?. EDBT: 602–605.
- Luckham, D. (2002). The power of events (Vol. 204). Reading: Addison-Wesley.
- Murthy, P., Bharadwaj, A., Subrahmanyam, P., A., Roy, A., RajanCloud, S., (2014). Big Data Taxonomy. Cloud Security Alliance's big data Working Group
- Nyce, C., (2007). Predictive Analytics White Paper. American Institute for Chartered Property Casualty Underwriters/Insurance Institute of America, p. 1. Available online at: <http://www.hedgechatter.com/wp-content/uploads/2014/09/predictivemodelingwhitepaper.pdf>
- OMG, (2011). UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems, June 2011. Version 1.1, OMG document: formal/2011-06-02

- Pearl, J. (2003). Causality: models, reasoning and inference. *Econometric Theory*, 19(675-685), 46.
- Quinton, C., Romero, D., Duchien, L., (2013). Cardinalitybased feature models with constraints: a pragmatic approach. In: *SPLC 2013: 17th International Software Product Line Conference*. Ed. by Tomoji Kishi, Stan Jarzabek and Stefania Gnesi. ACM, 2013, pp. 162–166. isbn: 978-1-4503-1968-3. doi: 10.1145/2491627.2491638.
- Quinton, C., Rouvoy, R., Duchien, L., (2012). Leveraging Feature Models to Configure Virtual Appliances. In: *CloudCP 2012: 2nd International Workshop on Cloud Computing Platforms*. ACM, 2012, pp. 21–26. isbn: 978-1-4503-1161-8. doi: 10.1145/2168697.2168699.
- Ranjan, R., Benatallah, B., Dustdar, S., Papazoglou, M.P., (2015). Cloud resource orchestration programming: overview, issues, and directions. *IEEE Internet Comput.* 19, 46–56
- Rivest, R., Shamir, A., Adleman, L., (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*. 21 (2): 120–126. doi:10.1145/359340.359342.
- Rodgers, J. L., Nicewander, W. A., (1988). Thirteen ways to look at the correlation coefficient. *The American Statistician*. 42 (1): 59–66. doi:10.1080/00031305.1988.10475524.
- Rossini, A., Kritikos, K., Nikolov, N., Domaschka, J., Griesinger, F., Seybold, D., Romero, D., (2015). D2.1.3 CAMEL Documentation. Available online at: [https://paasage.ercim.eu/images/documents/docs/D2.1.3\\_CAMEL\\_Documentation.pdf](https://paasage.ercim.eu/images/documents/docs/D2.1.3_CAMEL_Documentation.pdf)
- Tweed, R., James, G., (2010). A Universal NoSQL Engine, Using a Tried and Tested Technology. p.1 – 25, Available online at: <http://www.mgateway.com/docs/universalNoSQL.pdf>
- Valiant, L. G. (2011). A bridging model for multi-core computing. *Journal of Computer and System Sciences*, 77(1), 154-166.
- Veloudis, S., Verginadis, V., Patiniotakis, I., Paraskakis, I., Mentzas, G., (2016). Context-aware Security Models for PaaS-enabled Access Control. 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), Rome, Italy, April 23-25, 2016.
- Veloudis, S., Paraskakis, I., Petsos, C., Verginadis, Y., Patiniotakis, I., Mentzas, G., (2017). An Ontological Template for Context Expressions in Attribute-Based Access Control Policies. 7th International Conference on Cloud Computing and Services Science (CLOSER 2017), Porto, Portugal, April 24-26, 2017.
- Verginadis, Y., Michalas, A., Gouvas, P., Schiefer, G., Hübsch, G., Paraskakis, I., (2015). PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services. 5th International Conference on Cloud Computing and Services Science (CLOSER 2015), 20-22 May, Lisbon, Portugal, DOI 10.5220/0005489302060213
- Verginadis, V., Patiniotakis, I., Mentzas, G., (2016). D2.1 - Context-aware Security Model. PaaSword deliverable available online at: [https://www.paasword.eu/wp-content/uploads/2016/09/D2-1\\_Context-awareSecurityModel.pdf](https://www.paasword.eu/wp-content/uploads/2016/09/D2-1_Context-awareSecurityModel.pdf)

Yang, F., Huang, Y., Zhao, Y., Li, J., Jiang, G., & Cheng, J. (2017). The Best of Both Worlds: big data Programming with Both Productivity and Performance. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 1619-1622). ACM.

Youseff, L., Butrico, M., Da Silva, D., (2008). Toward a Unified Ontology of Cloud Computing, Grid Computing Environments Workshop (GCE08), held in conjunction with SC08.

Zahid, F., Verginadis, Y., Zolnierowicz, W., Skrzypek, P., Seybold, D., Kritikos, K., Mazumdar, S., Schwichtenberg, A., Domaschka, J., Horn, G., Gran, E., G., Baur, D., Masata, H., Gora, P., (2017). D2.1 System Specification. Melodic deliverable.

ZeroMQ, (2008). Broker vs. Brokerless. Whitepaper available online at: <http://zeromq.org/whitepapers:brokerless>

Ziv, J., Lempel, A., (1978). Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory. 24 (5): 530. doi:10.1109/TIT.1978.1055934.

## Appendix – Metadata Schema Serialization

We provide below an excerpt of the Metadata Schema serialized in XMI<sup>46</sup> that concerns the classes Big-Data Aspects, Data Density, Volume, Partitions, Velocity, Near Real-time Feed, Real-time Feed, On Demand Feed, Time-Series and their respective properties from the Big-Data model part of the Metadata Schema. The serialization used was decided based on the need for storing this vocabulary and any updates on it, in a Connected Data Objects (CDO<sup>47</sup>) server in the most appropriate way for seamlessly weaving new concepts in the CAMEL model. We note that the reader may find the serialization of the complete model here:

<https://bitbucket.7bulls.eu/projects/MEL/repos/metadata-schema/browse/>

```
<?xml version="1.0" encoding="UTF-8"?>
<mms:MmsConcept xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
  xmlns:mms="http://www.melodic.eu/metadata"
  name="Melodic Model" id="a2a76c78-c50f-4c96-b6e2-c60a33e13dfa"
  uri="mms:a2a76c78-c50f-4c96-b6e2-c60a33e13dfa"
  description="Melodic Metadatata Model"
  topLevel="true">
  <concept id="3f4853ea-0bbf-43cc-9a40-7cbb1494489c"
    uri="mms:3f4853ea-0bbf-43cc-9a40-7cbb1494489c"
    name="Big-Data Model" description="Big-Data Model">
    <concept id="210facdb-6c28-4993-bdbb-4b24f01e499c"
      uri="mms:210facdb-6c28-4993-bdbb-4b24f01e499c"
      name="Big-Data Aspects" description="Big-Data Aspects">
      <concept id="7e039d69-1fbb-4237-874c-5215c72198be"
        uri="mms:7e039d69-1fbb-4237-874c-5215c72198be"
        name="Data Density" description="Data Density">
        <concept id="ff5ebaa1-79ac-4151-86ef-8a768aa3d3bb"
          uri="mms:ff5ebaa1-79ac-4151-86ef-8a768aa3d3bb"
          name="Volume" description="Volume">
          <properties id="9ab918a7-6e8f-4d03-b5f5-6abc116108d2"
            uri="mms:9ab918a7-6e8f-4d03-b5f5-6abc116108d2"
            name="hasSize" description="hasSize"
            rangeUri="xsd:double"/>
          <properties id="cbff4ab8-038a-4f08-b7dd-accfdde8bdb5"
            uri="mms:cbff4ab8-038a-4f08-b7dd-accfdde8bdb5"
            name="fitsToMemory" description="fitsToMemory"
            rangeUri="xsd:boolean"/>
          <properties id="567c372b-4bd8-4277-87aa-f1731946a026"
            uri="mms:567c372b-4bd8-4277-87aa-f1731946a026"
            name="canBePartitioned" description="canBePartitioned"
            rangeUri="xsd:boolean"/>
          <concept id="e0acfb29-9ebc-44b9-934b-f7efae48806a"
            uri="mms:e0acfb29-9ebc-44b9-934b-f7efae48806a"
            name="Partitions" description="Partitions" />
        </concept>
      <concept id="c1e7ea6d-c892-42f1-a146-8c7497fd4bb6"
        uri="mms:c1e7ea6d-c892-42f1-a146-8c7497fd4bb6"
        name="Velocity" description="Velocity">
```

<sup>46</sup> <http://www.omg.org/spec/XMI/>

<sup>47</sup> <https://www.eclipse.org/cdo/>

```

<properties id="d2d6d093-e906-4c3e-8a26-f3af588e78fc"
  uri="mms:d2d6d093-e906-4c3e-8a26-f3af588e78fc"
  name="isContinuous" description="isContinuous"
  rangeUri="xsd:boolean"/>
<concept id="63477aec-6f48-402f-b1b5-de4526ecb66f"
  uri="mms:63477aec-6f48-402f-b1b5-de4526ecb66f"
  name="Near Real-time Feed"
  description="Near Real-time Feed" />
<concept id="17b7a83f-e46b-45c4-b849-4a9a03c6e19d"
  uri="mms:17b7a83f-e46b-45c4-b849-4a9a03c6e19d"
  name="Time-Series" description="Time-Series" />
<properties id="5749b271-e4d3-48a3-9167-dcbf38856ac7"
  uri="mms:5749b271-e4d3-48a3-9167-dcbf38856ac7"
  name="isOutputVelocity" description="isOutputVelocity" />
<properties id="5eed7811-520d-42cb-ad0b-dcd1c4e781cf"
  uri="mms:5eed7811-520d-42cb-ad0b-dcd1c4e781cf"
  name="isInputVelocity" description="isInputVelocity" />
<properties id="61127bd0-3ff1-4963-832f-5000cc3a3372"
  uri="mms:61127bd0-3ff1-4963-832f-5000cc3a3372"
  name="hasRate" description="hasRate" />
<concept id="6e12d39b-a606-429b-9f23-7facb6a49776"
  uri="mms:6e12d39b-a606-429b-9f23-7facb6a49776"
  name="Real-time Feed" description="Real-time Feed" />
<concept id="3152b991-722b-4b80-9fd8-8690dbf29a0e"
  uri="mms:3152b991-722b-4b80-9fd8-8690dbf29a0e"
  name="On Demand Feed" description="On Demand Feed" />
  </concept>
</concept>
.....
</concept>
.....
</concept>
.....
</mms:MmsConcept>

```