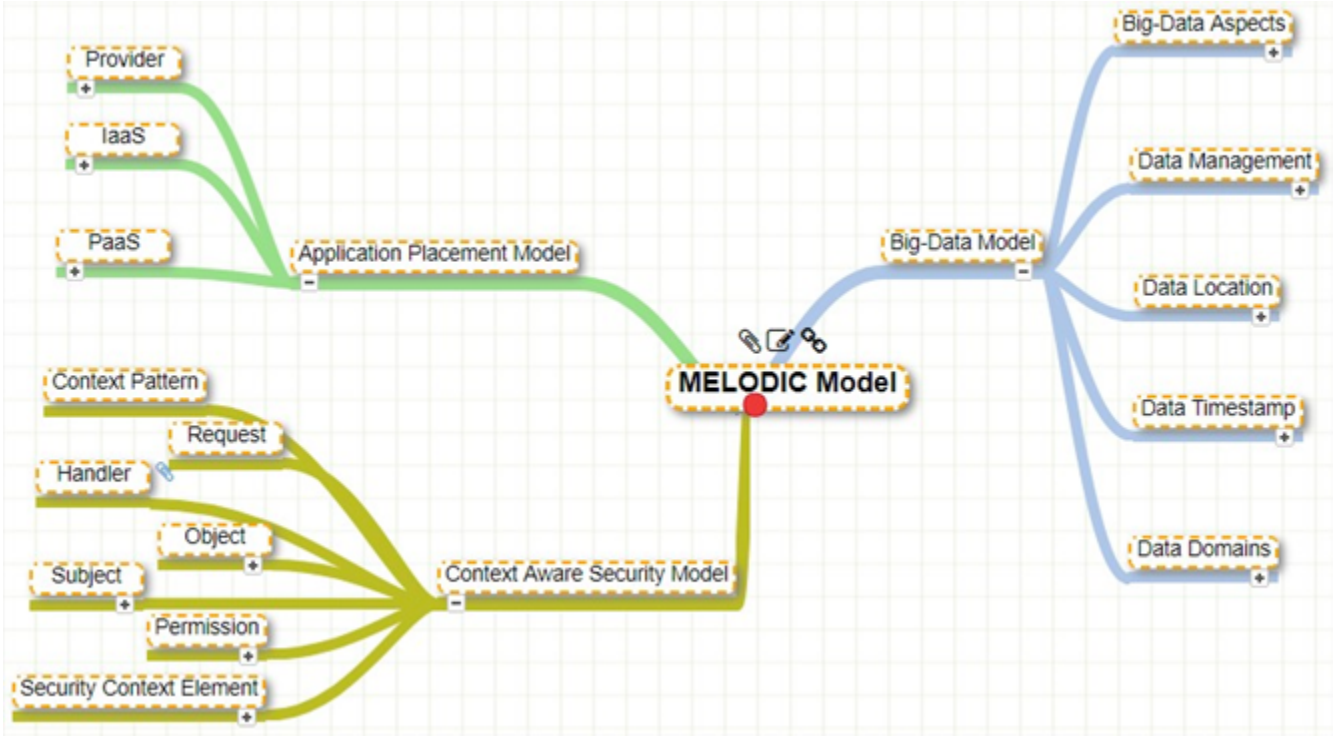
ii. How to update the Melodic Metadata Schema (Yiannis)

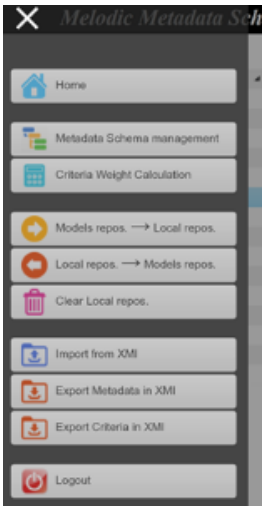
Melodic's Metadata Schema provides a vocabulary though a hierarchical structure of all the concepts (represented as lexical terms) that are relevant for describing cloud application requirements, big data aspects and characteristics (with respect to the input and output of these applications) and the offered cloud infrastructure capabilities for discovering optimised multi-clouds placement opportunities. In addition, it encapsulates all the necessary concepts for enabling the context-aware authorization functions that the Melodic platform supports. 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. For example, if we need to know the geographical location where processing of a certain big-data type will take place, then a relevant geographical processing location concept should be defined, along with its related instances (e.g., EU, GR, ASIA) and properties (e.g., latitude, longitude). This Schema is used for formally extending the CAMEL language if need. Concepts from this Schema potentially affect the Requirement, Metric, Scalability, Location, Provider and Security sub-models of CAMEL. A bird's eye view of the schema can be found in the following figure, while a 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 Also, a detailed explanation of all the classes and properties included in this Schema is provided here: https://melodic.cloud/wp-content/uploads/2019/01/D2.4-Metadata-schema.pdf



It is expected that the Metadata Schema will be adapted to the specific business and technical requirements of each application of a single organisation. For this purpose, MUSE - a Metadata Schema editor has been developed for creating, updating and maintaining the Metadata Schema. Typically, this task is undertaken by the administrator. Below we show how MUSE can be used.

Initialization

Before using the editor, it is necessary to fetch the Metadata Schema from the Models Repository into the Local datastore. This is achieved by clicking on the "Models repos. Local repos." menu item. The menu appears by clicking the hamburger style glyph at the upper left corner of the page as seen in the next Figure that illustrates the editor menu.

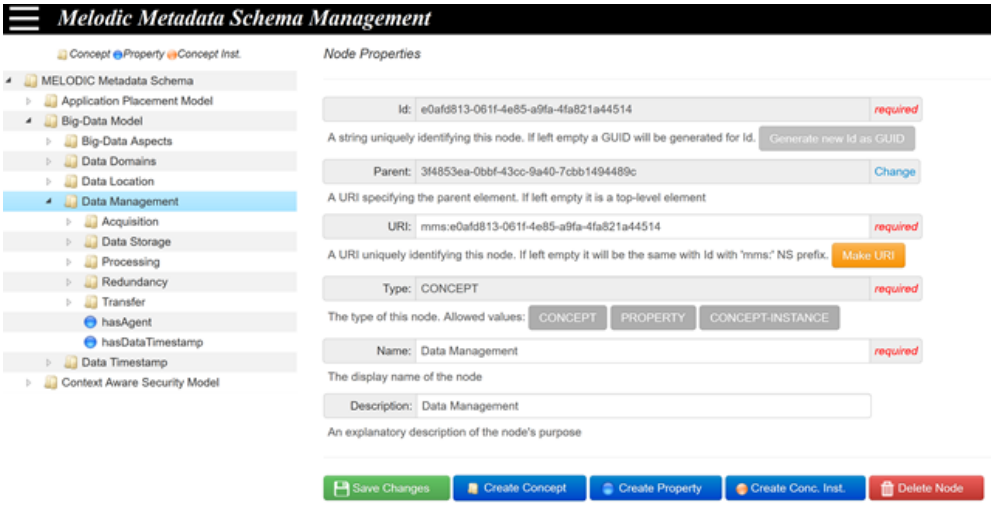


From the menu, the user can also transfer any changes made in the Metadata Schema, from the Local Repository into the Models Repository (menu item "Local repos. Models repos.") thus making them visible to the rest of the Melodic components. Other options include emptying the Local Repository (menu item "Clear Local repos."), exporting Metadata Schema from Models repository (in XMI format), and reversely importing XMI files into the Models Repository.

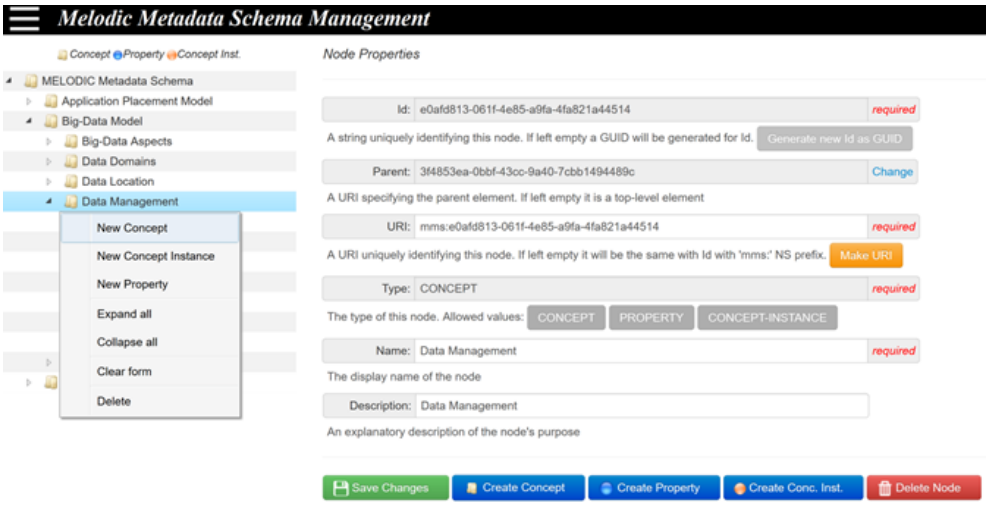
Updating/Instantiating the Schema

By selecting "Metadata Schema management" from the menu, the Metadata Schema editor web page is loaded (as seen in the next figure). Using the tree-view in the left-hand side of the page, the user can navigate through the Metadata Schema. Concepts are represented with yellow folder icons in the tree view, whereas other artifacts (e.g. concept properties) are represented with coloured balls. By selecting a tree node, its details are loaded into the "Node Properties" form, found in the main area of the page. Furthermore, the form fields change according to the type of the tree node selected (i.e. concept, concept property or concept instance). For example, the Range form field is only available for the concept property nodes. The white text fields can be edited while the grey ones are read-only. At the bottom of the page, there is a row of buttons for creating new nodes (concepts, properties, and instances), deleting the selected node or saving changes in the form.

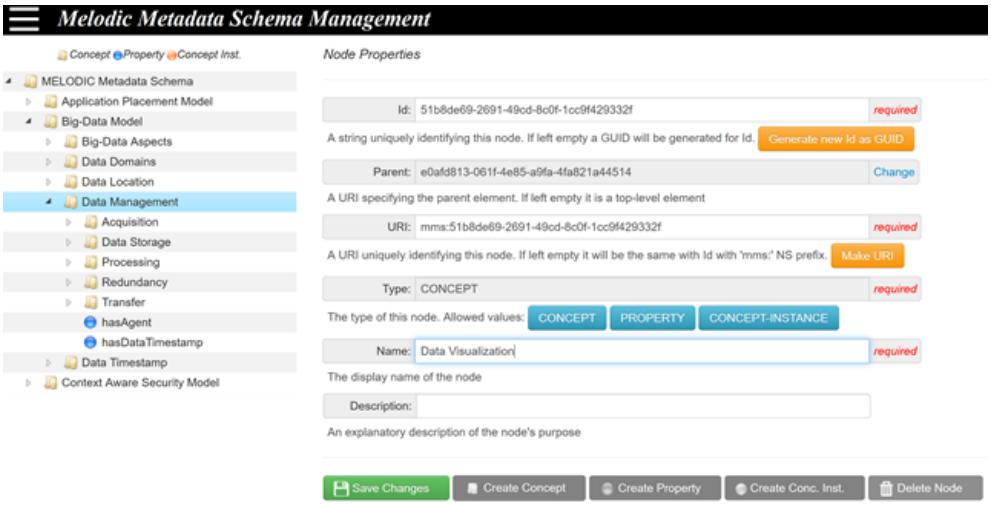
In the remaining section, a simple use case, where a new sub-concept will be created, is detailed. First, the user selects the "Data management" node in the tree view, as shown in the Figure below. This node will be the (immediate) parent concept of the concept to be added. Upon selection, the parent node' s data are fetched from the Local Repository and are displayed in the "Node Properties" form. Next, the "Create Concept" button must be pressed to start creating the new sub-concept.



The same functionality is also available through a context popup menu, which is available by right clicking on the parent node, and then clicking the "New Concept" item (as seen in the next Figure).



Both actions (i.e. pressing “Create Concept” or clicking “New Concept” in context popup menu) will make the “Node Properties” form (in the main page area) to adapt in order to include only those fields that are relevant to Concepts (i.e. Id, Parent Id, URI, Type, Name, Description). The values of Id and URI fields are automatically completed, as shown in next Figure.



The user can modify Id and URI values if needed, fill-in the new concept’s name, for instance “Data Visualization”, and optionally give a description, in the corresponding fields. Eventually, by pressing the “Save Changes” button, the new sub-concept’s data are submitted to the Local Repository for storing. Afterwards, the tree view is refreshed in order to include the newly added concept.