LinkEHR-Ed

Archetype Editor
v0.8

User's Manual

Biomedical Informatics Group – IBIME
ITACA Institute
Universidad Politécnica de Valencia

1. About us

The Biomedical Informatics (IBIME) area of the Institute for the Applications of Advanced Information and Communication Technologies (ITACA) consists of 20 members and it is coordinated by Ph. D. Montserrat Robles Viejo.

ITACA is a research and development centre of the Universidad Politécnica de Valencia (UPV), Spain, whose purpose is to promote and carry out research, technological development and transference of results in the field of information and communication technologies.

The research of IBIME-ITACA is concerned with the use and development of methods and tools for the acquisition, processing and management of biomedical data and knowledge. The research is characterised by its multidisciplinarity and close linkage with health professional and institutions. IBIME collaborates with several national and international groups working on bioinformatics and biomedical informatics mainly in Europe and Canada due to European and National projects and research stays. The area combines successfully four main fields of biomedical informatics, namely, health information engineering, pattern-recognition, medical imaging and bioinformatics.

The Health Information Engineering area of IBIME has been working since 1999 in the field of the efficient use of information and communication technologies for the management of biomedical information. Current lines of work are:

- Standardization and modelling of electronic clinical history by means of ontologies and archetypes of sanitary information
- Semantic interoperability of health information systems
- Integration of biomedical data
- Modelling and management of phenotypic information
- Visualization of biomedical information

More info:

About LinkEHR-Ed: http://pangea.upv.es/linkehr
About IBIME: http://www.ibime.upv.es
About ITACA: http://www.itaca.upv.es
About UPV: http://www.upv.es

Contact Information: You can contact us for any suggestions or bug notifications through the project web page (see above) or by sending an email to:

José Alberto Maldonado	David Moner	Diego Boscá
jamaldo@upv.es	damoca@upv.es	diebosto@upv.es

2. About LinkEHR-Ed

LinkEHR-Ed is an integration editor which allows mapping an archetype to existing data sources and generates the XQuery scripts that will extract and transform data from the sources into a standardized XML document.

It is a result of a research project funded by the Spanish Ministry of Education and Science, reference: TSI-2004-06475-C02 and TSI2007-66575-C02.

3. LinkEHR-Ed releases

Previous versions:

- LinkEHR-Ed v0.1: Just an ADL editor.
- LinkEHR-Ed v0.5: Archetype editor independent of the underlying reference model used for defining attributes. This is the open source version.
- LinkEHR-Ed v0.5.1 to LinkEHR-Ed v0.5.4: Minor changes and bugs resolved.
- LinkEHR-Ed v0.8: Current release includes the integration archetypes feature. Only for XML data sources.

Future versions:

- LinkEHR-Ed v0.9: This release will include support for relational data sources.
- LinkEHR-Ed v1: First final and usable version of the complete LinkEHR-Ed

4. What is LinkEHR-Ed?

LinkEHR-Ed is a visual tool implemented in Java under the Eclipse platform which allows the edition of archetypes based on different reference models, the specification of mappings between archetypes and data sources and the semi-automatic generation of data conversion scripts which translate unnormalized data into XML documents which conform to the reference model and at the same time satisfy the data constraints imposed by archetypes.

LinkEHR-Ed explores the use of archetypes as a means to achieve standardization and semantic data integration of distributed health data. The main objectives are twofold.

Firstly, in the context of data integration, to use of archetypes as a semantic layer over the data repositories, whose contents need to be integrated and exchanged, associating them with formal semantics. As aforementioned, the main purpose of archetypes is to describe information in the form of a set of machine-processable domain concept definitions based on a reference model. Therefore, archetypes may act as semantic descriptions that capture the information contents of heterogeneous repositories.

Secondly, we intend to employ archetypes for making public existing clinical information in the form of standardized EHR extracts. For this purpose, we take advantage of their data definition facet, which was formalized in the previous section. Archetypes explicitly specify the structure and content of valid instances of the underlying reference model to which interchangeable data instances must conform. Thus, it becomes necessary to transform data from the local sources with a particular structure or schema to meet the data structures defined by archetypes. This problem is known in the literature as the data exchange (translation or transformation) problem. Data exchange requires at the schema level an explicit representation of how the source and target schemas are related to each other; these explicit representations are called mappings. Mappings between two schemas can be specified in different ways. They can be a query expressed in SQL or XQuery, a predicate in first order logic, a set of correspondences each of which relates a element from the source schema to an element from the target schema or even a third schema that relates the other two via two sets of correspondences, i.e. a reified mapping. At the data level, data transformations operate on the instances of the schemas rather than on the schemas themselves, they transform source instances into target instances according to the mapping defined between the corresponding schemas. Data transformations may be expressed in a specific data transformation language, such as XQuery or by using a general purpose language as Java.

Since the health data to be made public resides in the underlying data sources, it is necessary to define some kind of mapping information that links entities described in the archetype (object nodes and attributes) to data elements in data repositories (e.g. elements and attributes in the case of XML documents, tables and attributes in the case of relational data sources). We use the term integration archetype to denote an archetype for which a mapping specification to a set of data sources has been defined, i.e.:

Integration archetype = archetype + mapping specification.

An integration archetype can be considered to be a view that provides abstraction in interfacing between the data sources that hold the data to be shared and the reference model used to communicate these data in the form of standardized EHR extracts. It is necessary to remark, that there exits a one to many relationship between archetypes and integration archetypes. Given an archetype, there may be different mappings, one for each different setting that wishes to use the archetype to describe and share its data. LinkEHR-Ed is a visual tool for defining integration archetypes.

Although LinkEHR-Ed is oriented to the construction of integration archetypes it may operate as a pure archetype editor. It can load ADL files and generate both ADL and XML according to the XML schema defined by the Consortium OpenEHR. At its core lies the Archetype Object Model an object oriented model for archetypes, this has also been adopted by CEN/TC251 EN13606. It uses the Java implementation of the AOM and the ADL parser developed by ACode, although several addition have been made in order to satisfy extra requirements such a multi reference model support and mapping to data sources.

With LinkEHR-Ed new archetypes can be defined from scratch, for instance to describe the data structure and semantics of legacy data such as messages or database schemas. It is also possible to define new archetypes by specializing or altering existing ones, such as those drawn from a public available archetype repository. In any case, LinkEHR-Ed is intended to support the mapping to data sources.

***NOTE**: LinkEHR-Ed has two main uses:*
- As a multireference model editor of archetypes.
- As an integration archetype editor which allows the integration and standardization of legacy data.

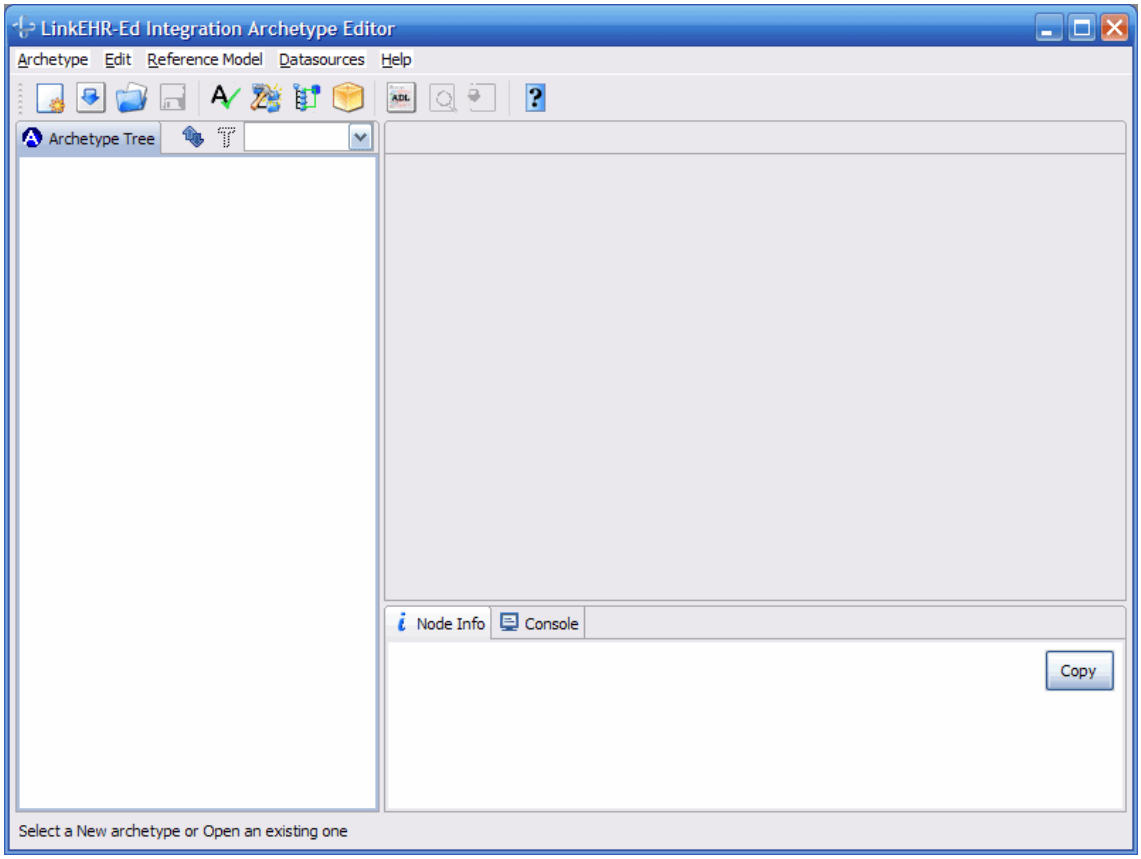
At this stage the first use is publicly available. The second is not open-sourced.

5. Archetype Edition: step by step tutorial

This guide will show up the basis of LinkEHR-Ed operation by showing you how to import both OpenEHR and CEN EN13606 reference models into LinkEHR-Ed. Later we will introduce how to open an archetype, a creation of a new archetype and the specialization of an archetype.

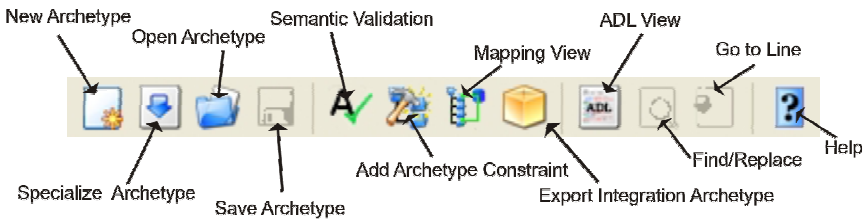
1. Running LinkEHR-Ed for the first time

The first time you open LinkEHR-Ed you will see this window



Capture 1: LinkEHR-Ed first run

LinkEHR-Ed toolbar has available the most common used actions. Capture 2 shows the meaning of buttons, although they should be auto explicative enough.



Capture 2: The meaning of the buttons on the toolbar

2. Configure LinkEHR-Ed

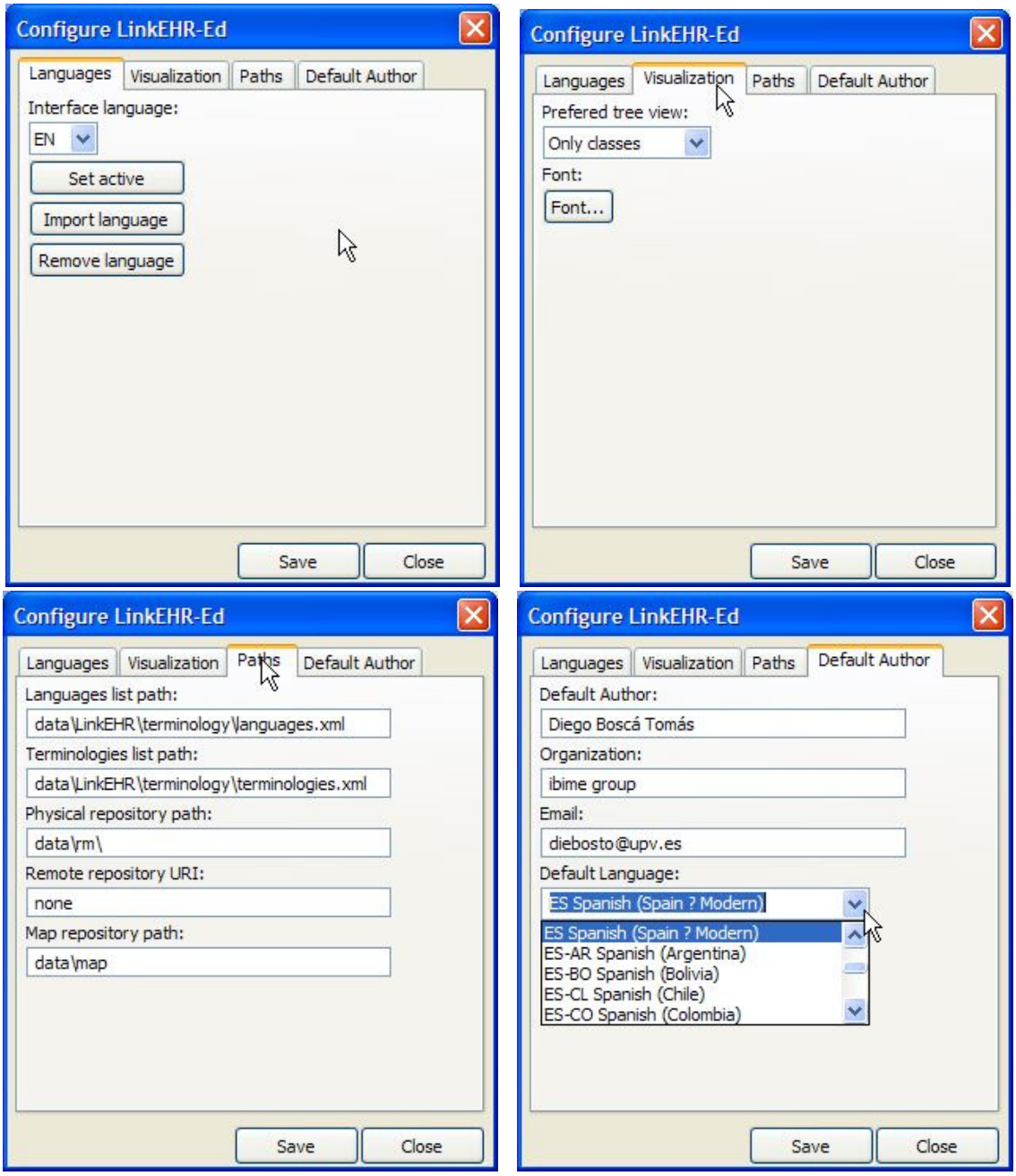
User preferences can be changed in LinkEHR-Ed's configuration dialog. This dialog is located at *Help* → *Configure LinkEHR-Ed*. The dialog has four tabs, each one controlling a part of LinkEHR-Ed (see capture 3).

a. **Languages tab**: controls the interface language of LinkEHR-Ed. Currently LinkEHR-Ed is only available in English, but we plan to release it on several languages. That's why the buttons have no effect at this moment.

b. **Visualization tab**: you can define the preferred visualization of the archetype tree (in order to show the ontology description, classes' names or both) and the default font for the archetype tree. This also can be configured by pushing the button  in the archetype tree view at any time during the archetype edition.

c. **Paths tab**: you can change the paths used by LinkEHR-Ed, such as the internal (physical) repository path and the location of language and terminology XML files.

d. **Default author tab**: you can insert the default information of the author in order to avoid retyping it everytime you create a new archetype. The Default language refers to the primary language selected by default when creating a new archetype through the "Create new archetype" wizard.



Capture 3: Configuration of linkEHR-Ed

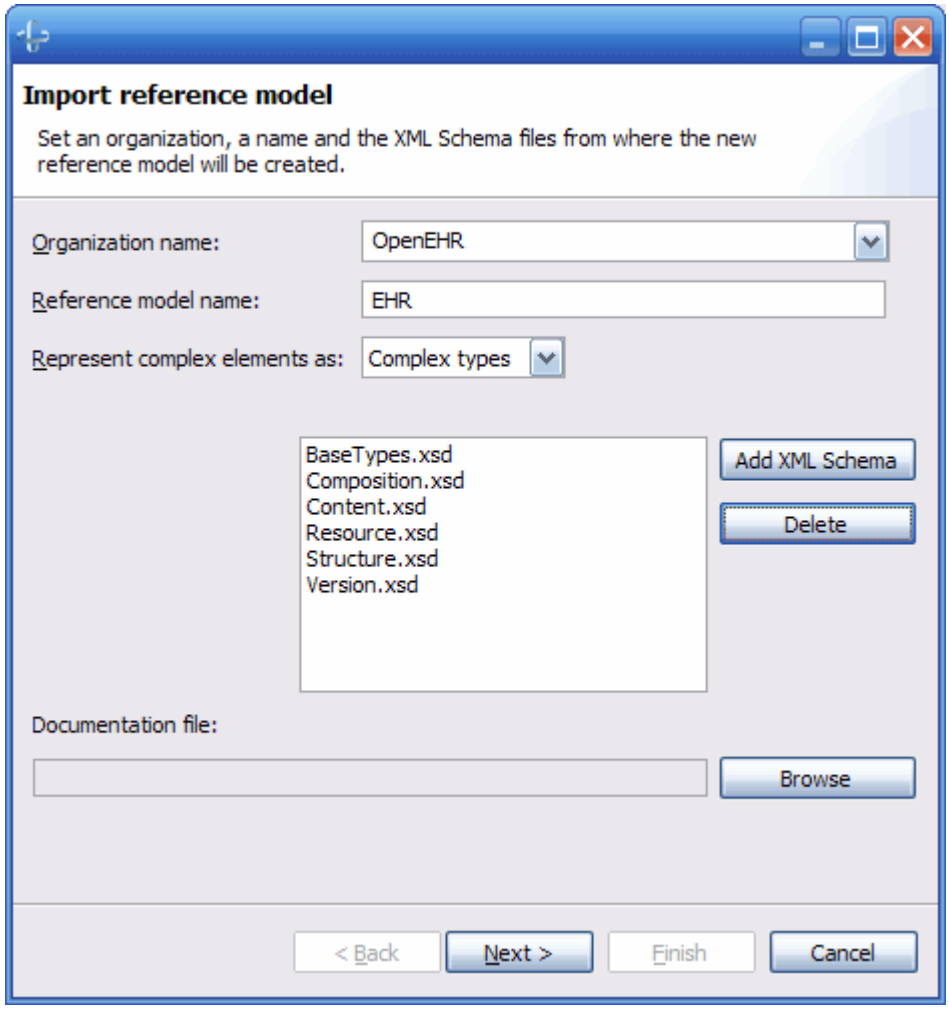
3. Adding the OpenEHR reference model to LinkEHR-Ed

To import a new reference model you have to open the import reference model wizard, which is located under *Reference Model* → *Import Reference Model*. There you can introduce the organization name, the model name and the XSD schemas defining the reference model, as you can see in capture 4.

Since we are constraining an OO model, the distinction between classes and attributes is crucial. In fact, different types of constraints are applied accordingly. For instance, classes

may be represented as elements or as type definitions. In the former approach, XML instances contain elements tagged with the class names while in the latter class names only appear in the schema. When importing a new reference model, it must be indicated whether or not classes are represented as elements in the schema. It is supposes that all classes are represented in the same way.

In the case of the OpenEHR official XML Schemas, you must set the combo "Represent complex elements as" to Complex types.



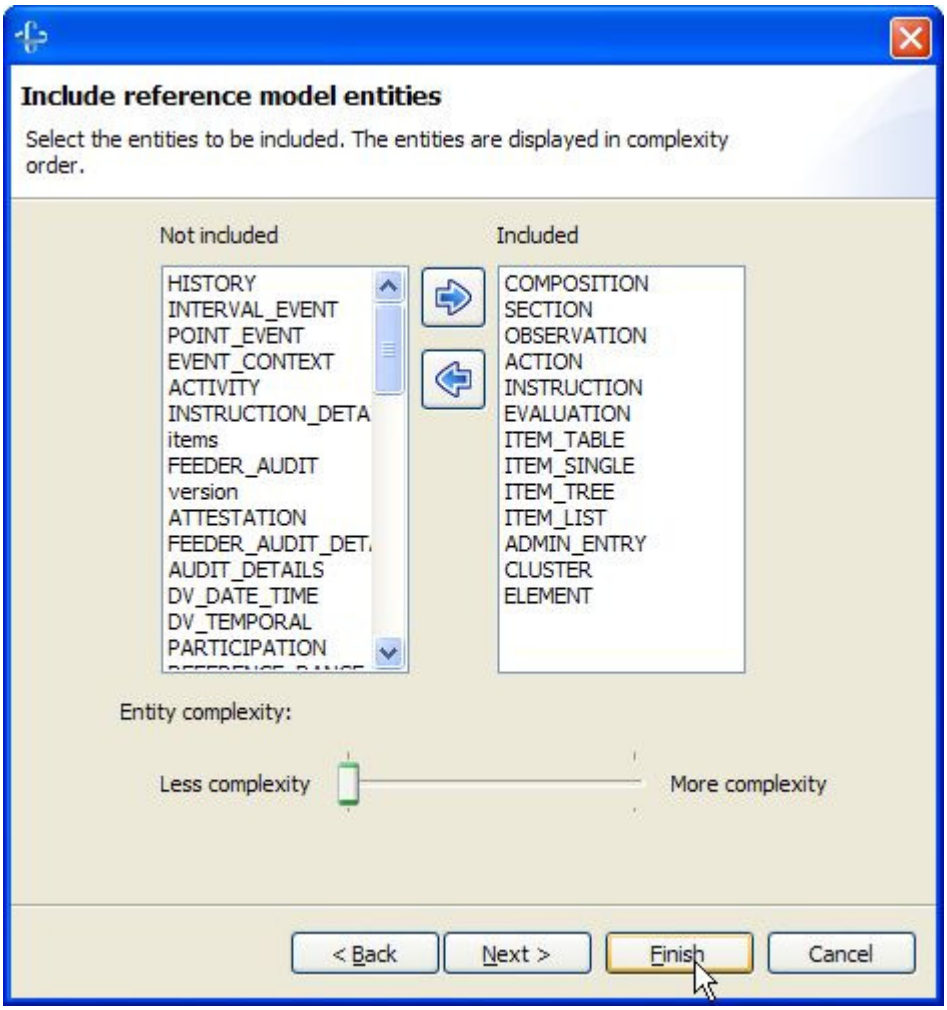
Capture 4: Importing OpenEHR reference model

Once all this information has been introduced you can push Next button. In the next wizard page will appear the list of entities available to be generated. The list is filled with the entities in complexity order, as it is likely that the more complex entities will be the ones basic to that model, also called Business Objects. If a needed entity does not appear in the list you can move the slider in order to reveal less complex entities. They are hidden at first to simplify the whole process.

When you have all the desired entities visible you only have to select them and click on the arrow to the right. This will show the selected entities on the "included" list.

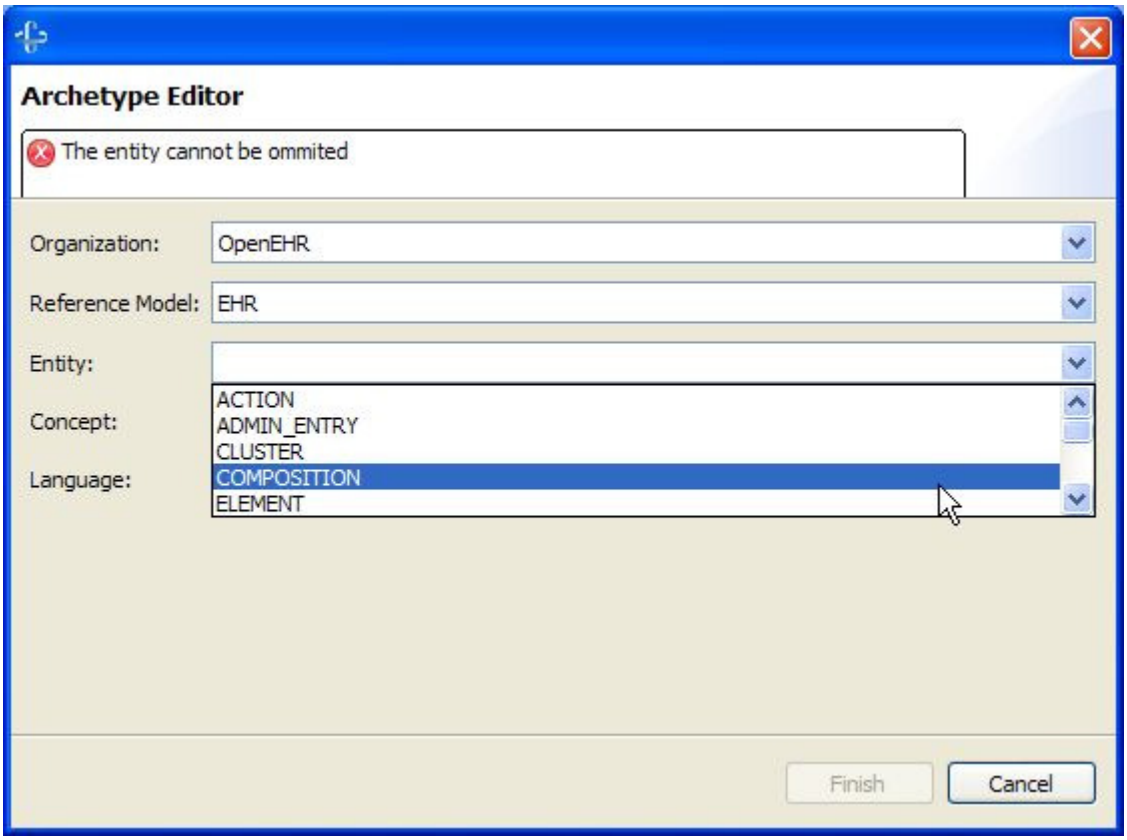
***NOTE**: Included entities will be the only ones available as basis for creating new archetypes at the* New archetype *dialog.*

When at least one entity is included you can finish the importation process. Capture 5 shows the entities of OpenEHR reference model.



Capture 5: The entities of OpenEHR reference model

Now we can check that the imported reference model has been included by trying to create a new archetype. Push *Archetype* → *New* and the window of capture 6 will appear. See section 5 for more details.

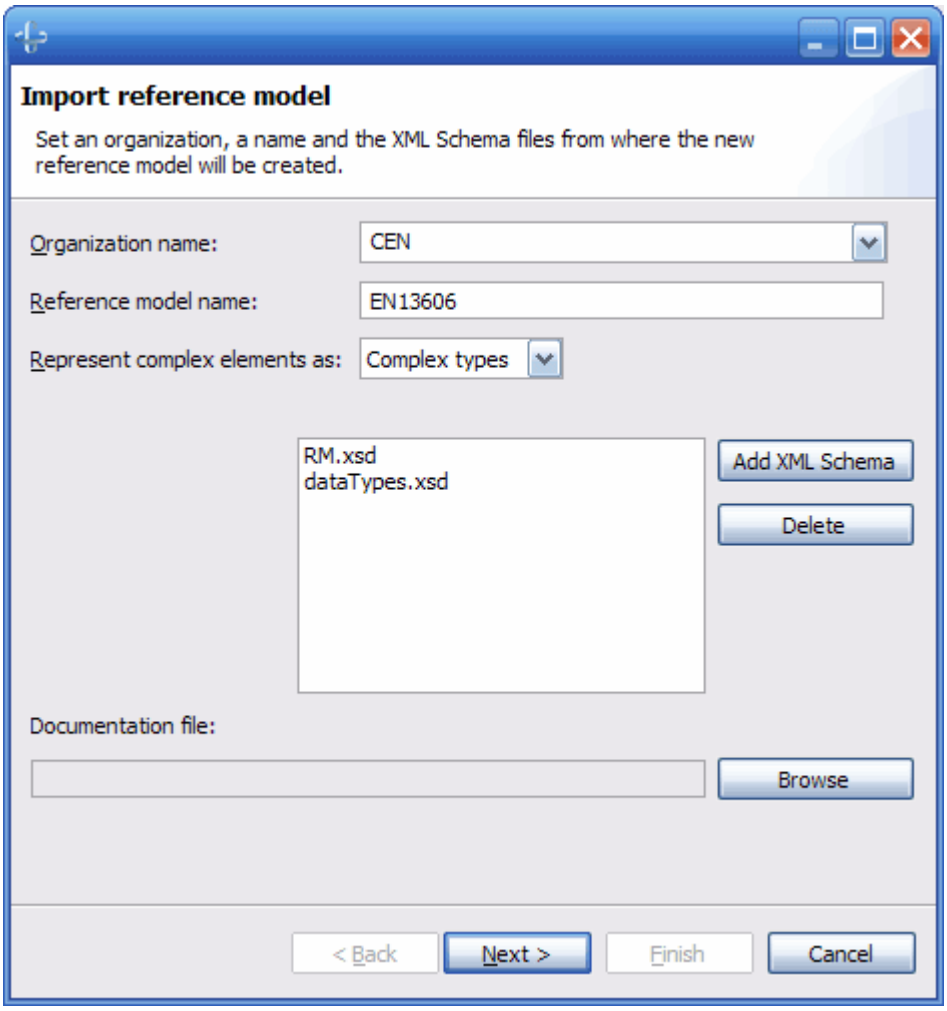


Capture 6: The new archetype wizard with the OpenEHR reference model included.

4. Adding CEN EN13606 reference model to LinkEHR-Ed

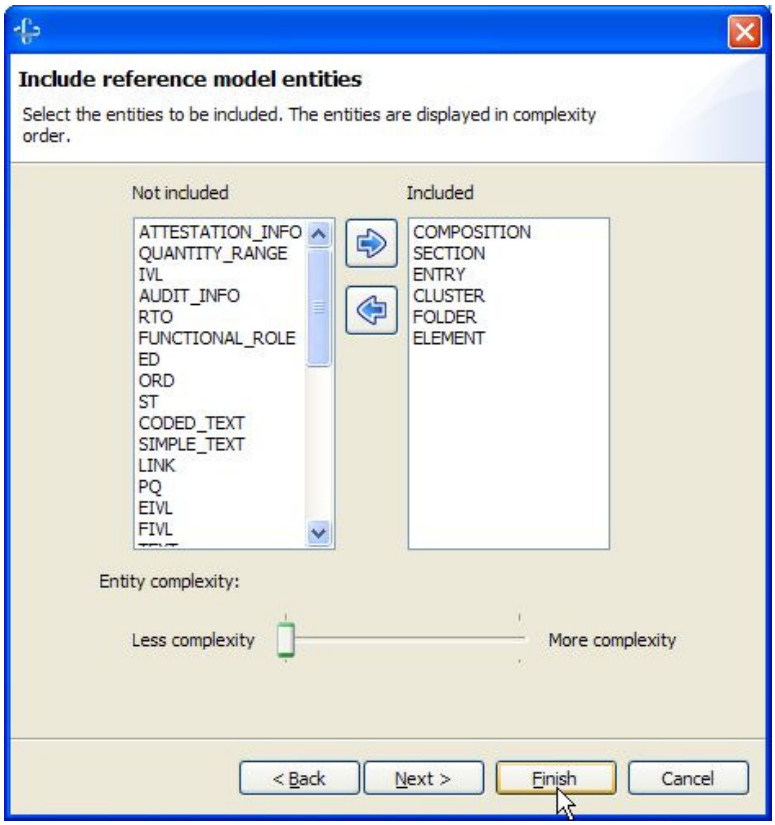
In the same way we have added the OpenEHR reference model we can add the CEN EN13606 reference model to LinkEHR-Ed. So the first page will be as shown on the capture 7.

> ***NOTE**: There is no official XML Schema for CEN EN13606. We have developed our own schema and it is available through the LinkEHR-ED project webpage.*



Capture 7: Importing CEN EN 13606

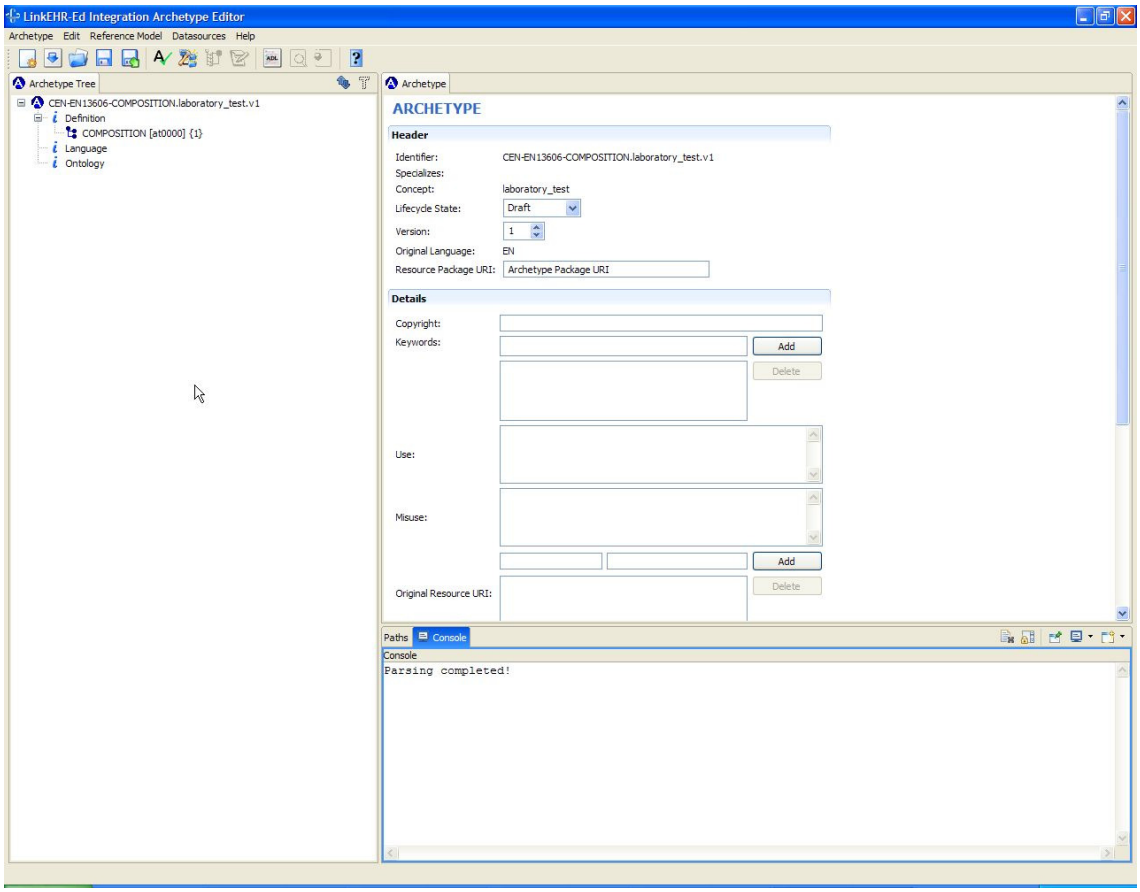
In the same way as before, we select the reference model entities to be included and push finish. The entities needed for EN13606 are shown at capture 8.



Capture 8: Reference model entities for CEN EN13606

5. Creating a new archetype

To create a new archetype we only have to push the New Archetype button of the toolbar or the menu *Archetype* → *New* and a wizard (shown at capture 6) will be opened. After selecting the organization, reference model, entity, concept and language a new and empty archetype will be loaded and ready for being edited (capture 9).



Capture 9: LinkEHR-Ed with a new archetype created.

6. Opening an archetype

This process is very straightforward. By pushing the Open button on the toolbar or the menu *Archetype* → *Open* a typical system open dialog will be shown. Choosing then an ADL file (version 1.4) will load it into LinkEHR-Ed.

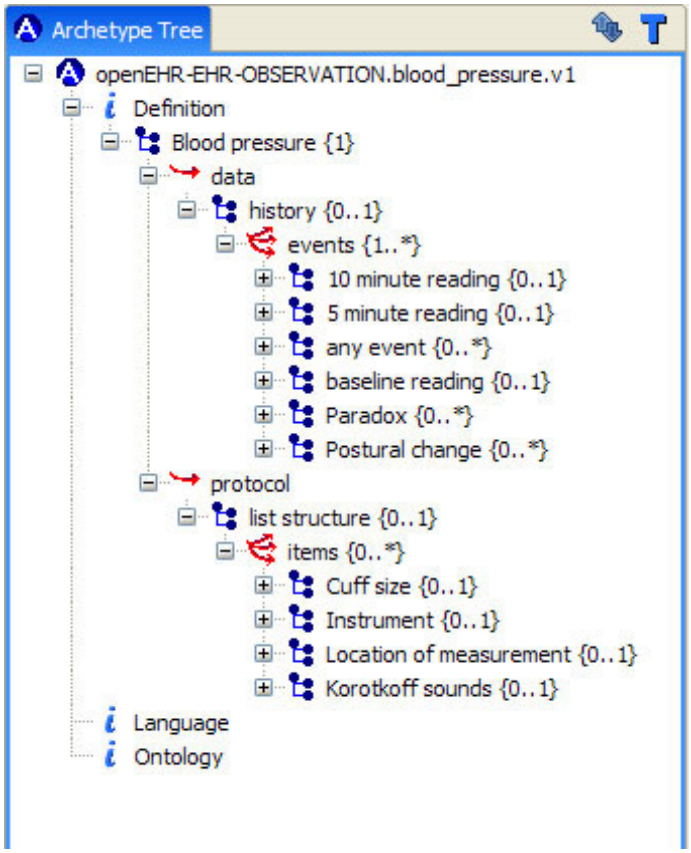
If the ADL is not valid, an error will be shown indicating the line where the error has been detected. Moreover, although the archetype tree can not be built the ADL code is available and can be accessed through *Go to ADL* button (represented with this icon ). Looking at the ADL code will allow you to easily detect the error, fix it and recompile the ADL text (by pushing the  button again) to generate the visual representation of the archetype.

7. Editing an archetype

The main window of LinkEHR-Ed has four main components or views:

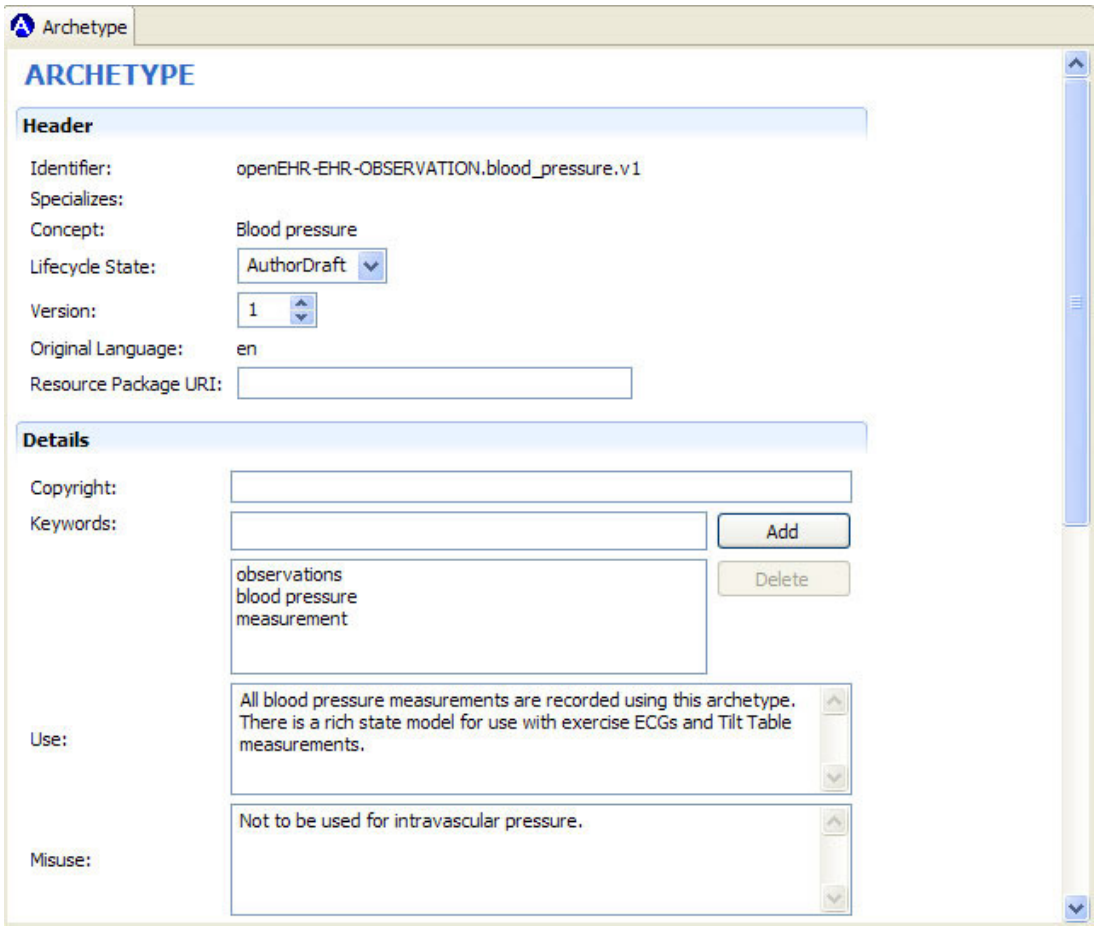
a. **Archetype tree view**: Here is where the archetype structure is represented as a tree, including its header and description, the definition tree, the language section and the

ontology section (capture 10). The button  switches the definition branch of the tree between technical and non-technical view.

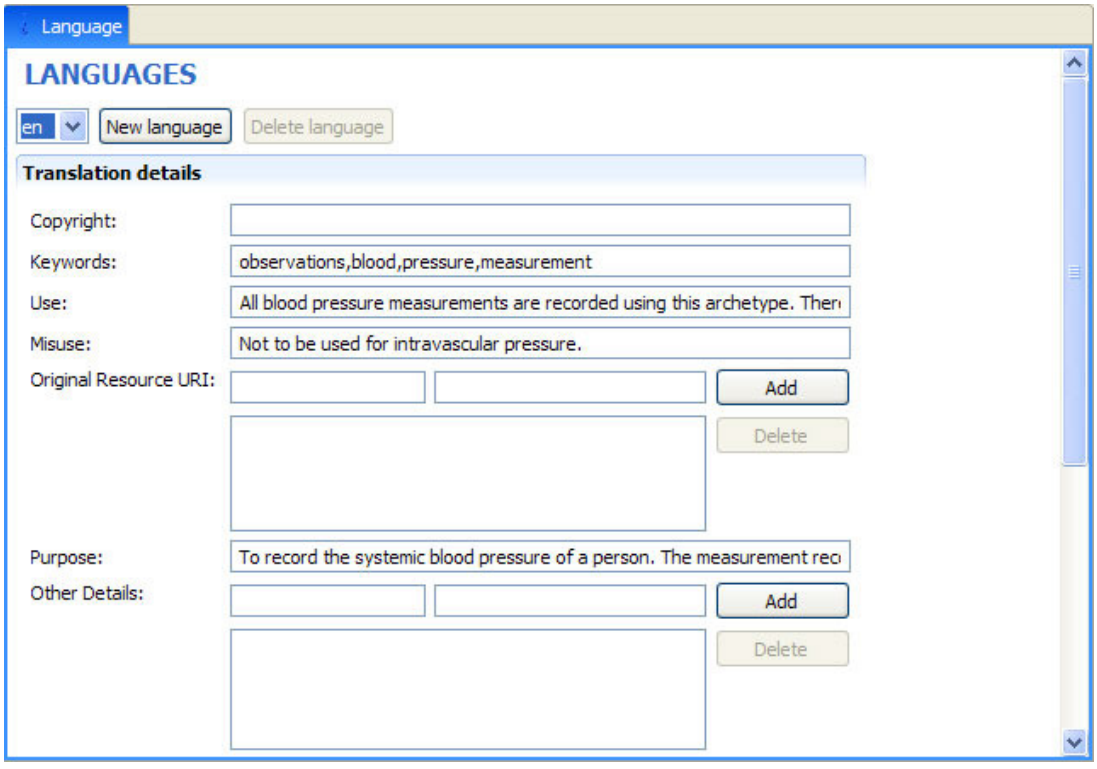


Capture 10: The archetype tree containing all of its structure

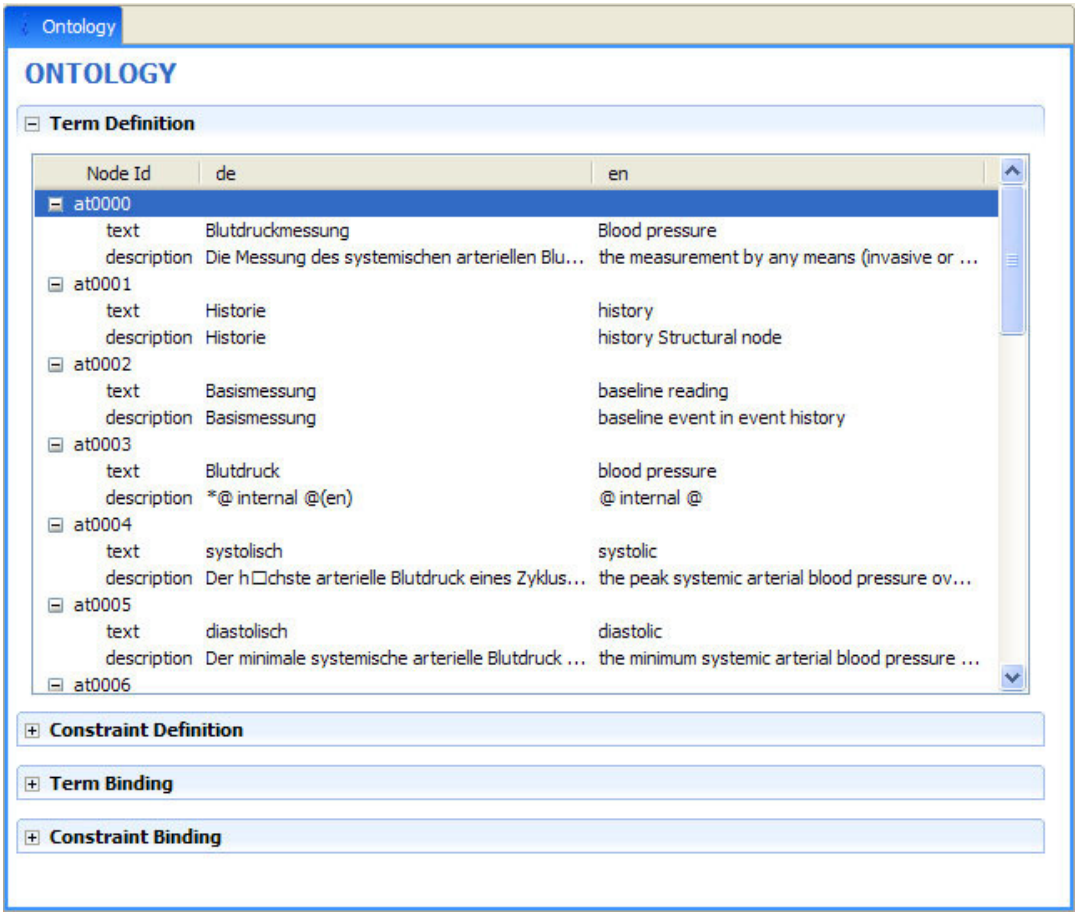
b. **Details view**: At the right side of the screen is where the different forms for introduction of data are loaded. A different form is associated to each kind of node of the archetype tree (see captures 11, 12 and 13).



Capture 11: The archetype header a description form

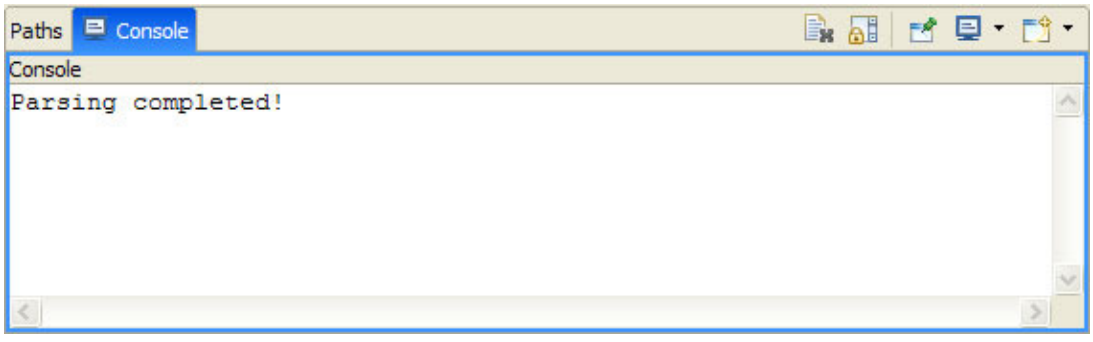


Capture 12: The language section form



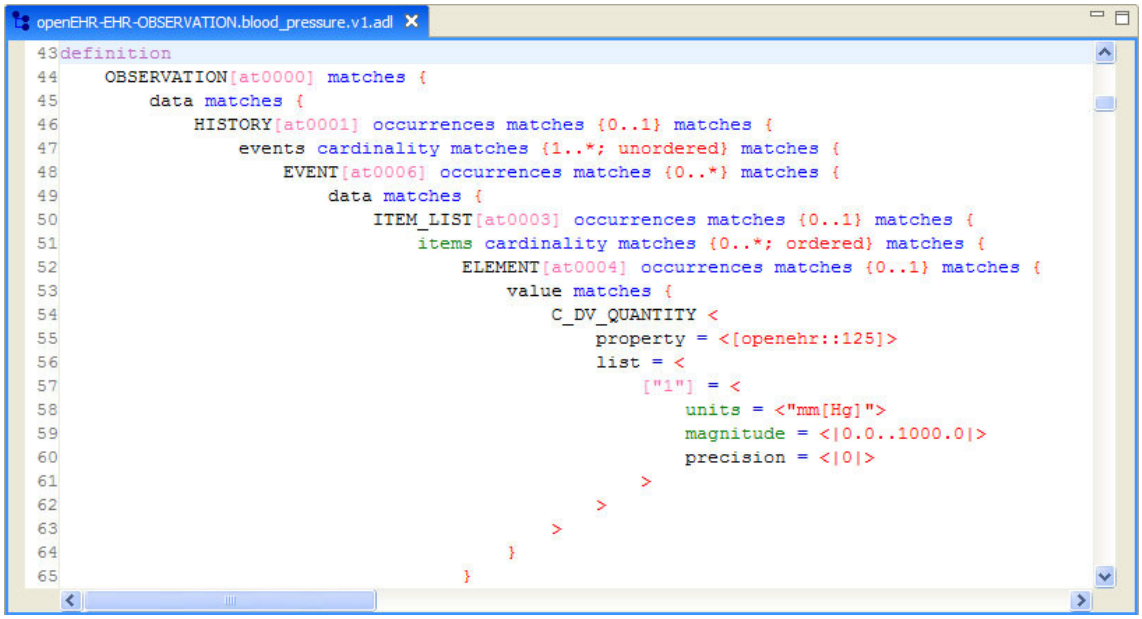
Capture 13: The ontology section form

c. **Console view**: This view shows the application messages and also has a tab to watch the path of the selected node of the definition tree.



Capture 14: The console view

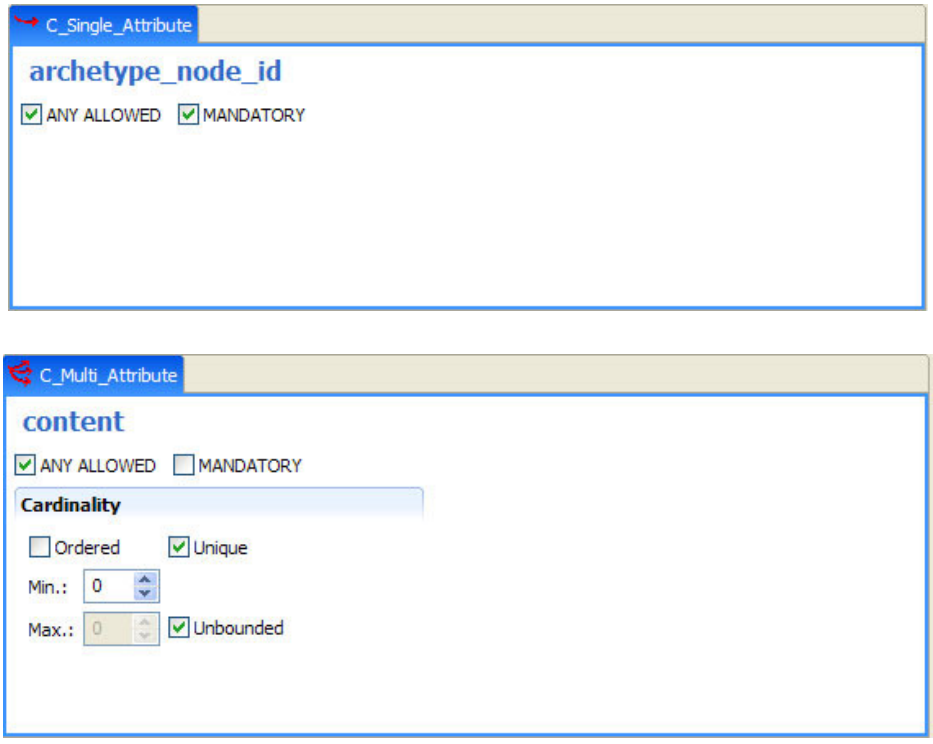
d. **ADL view**: Once an archetype has been loaded or created, you can always switch to the ADL view in order to see and edit it. Every change made in the ADL will be reparsed and validated before going back to the visual representation. You can switch to the ADL view by pushing the button,  and return to the previous view by pushing it again.



Capture 15: The ADL edition view

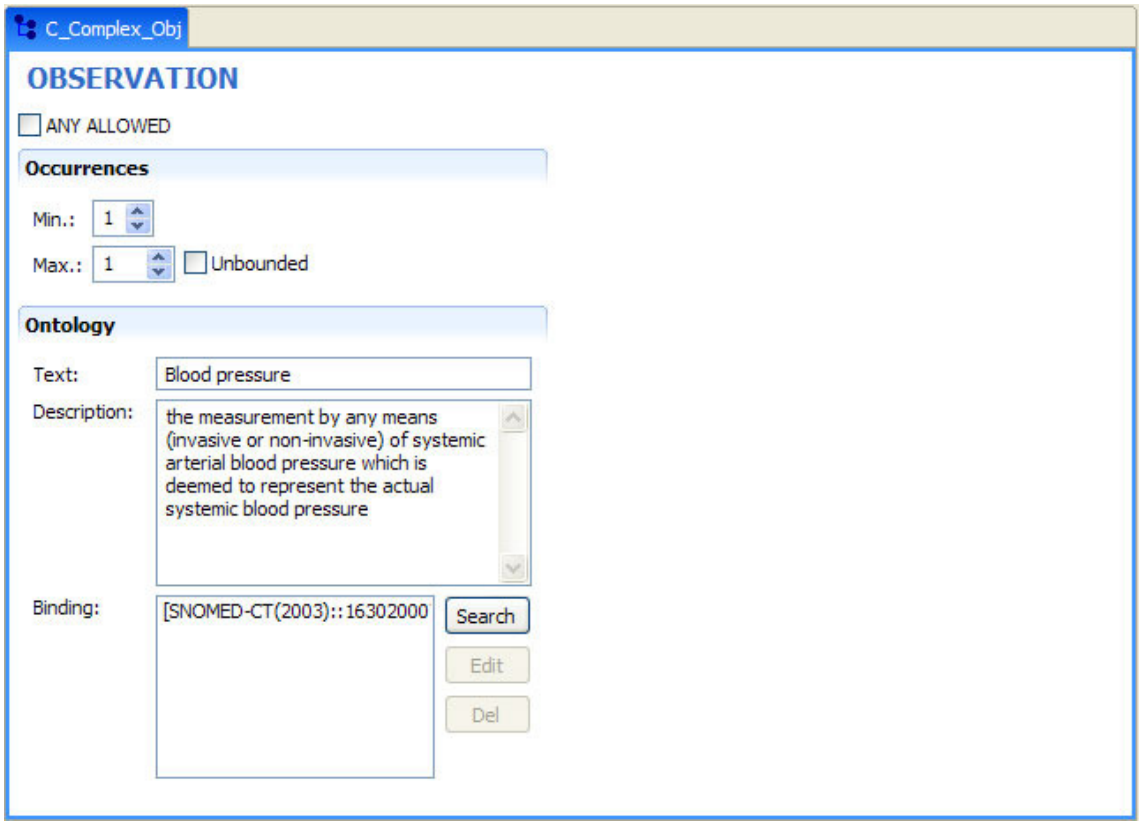
The main edition process is done at the definition branch of the archetype tree. By clicking on any node a form is shown in the details view with the editable information of the node. This form will depend on the type of node:

- **Attribute**: There are views for editing properties of Single and Multiple attributes.



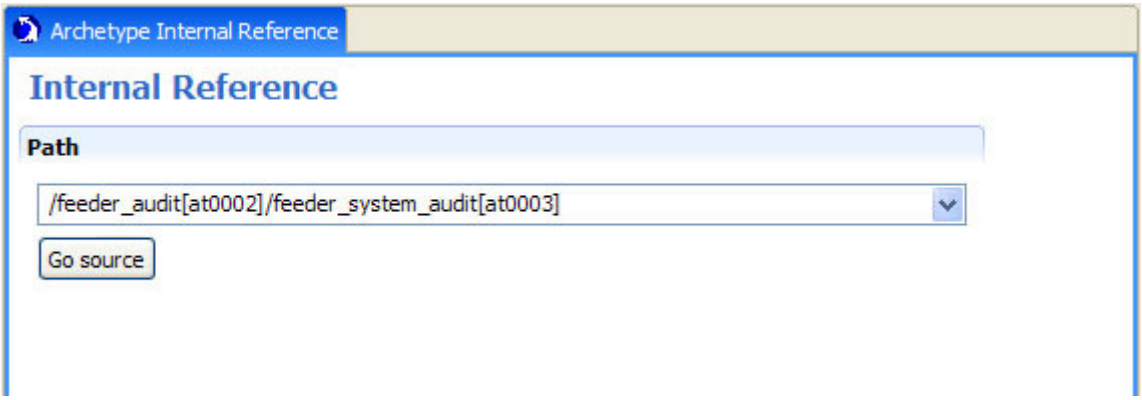
Capture 16: Single and multiple attributes view

- **Complex Object**: Occurrences of complex objects can be modified through their respective form. Their ontology information (term definition and term binding) can also be edited from this form.



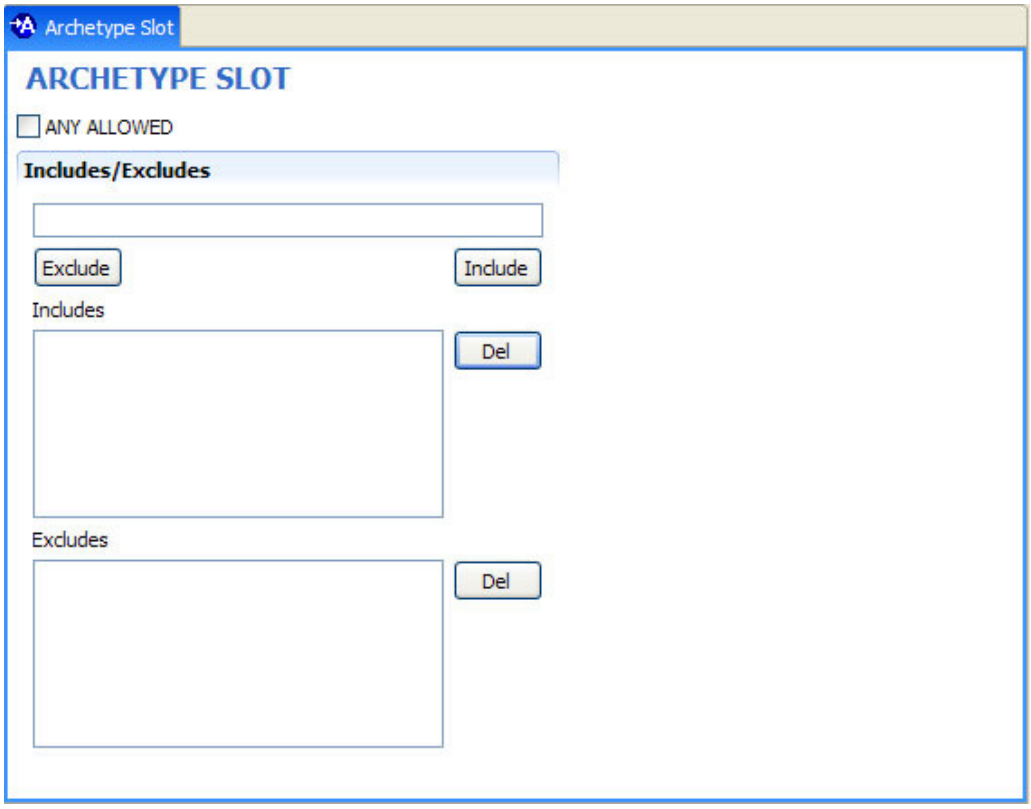
Capture 17: CComplexObject edition view

- **Internal Reference**: When an internal reference is created, a list of the available target nodes of the same reference model type is provided. You can change the target node at any time or directly jump to its definition.



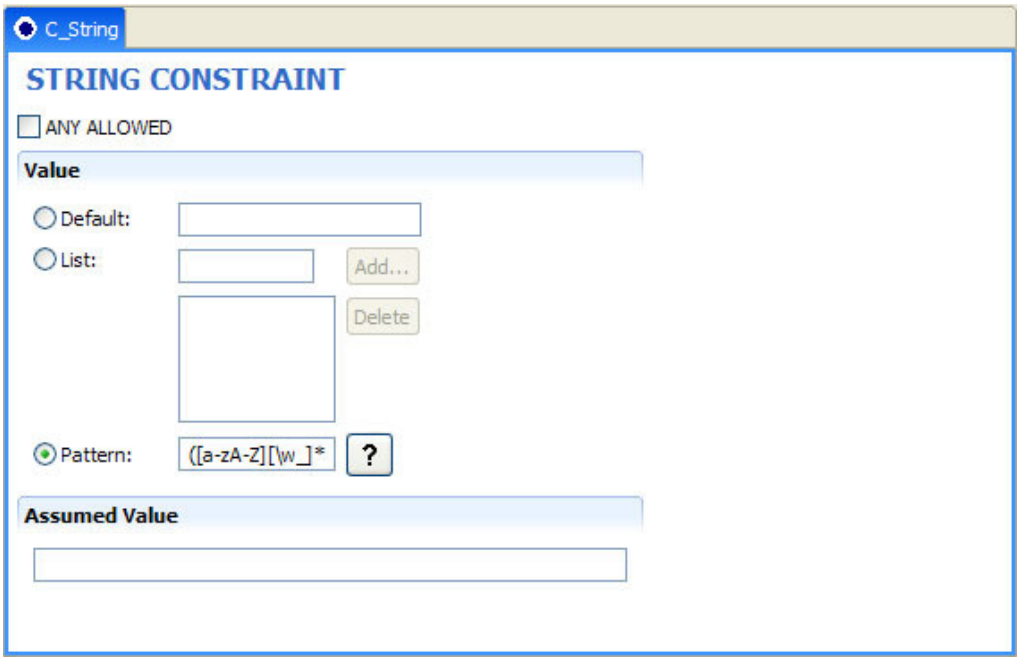
Capture 18: Internal reference view

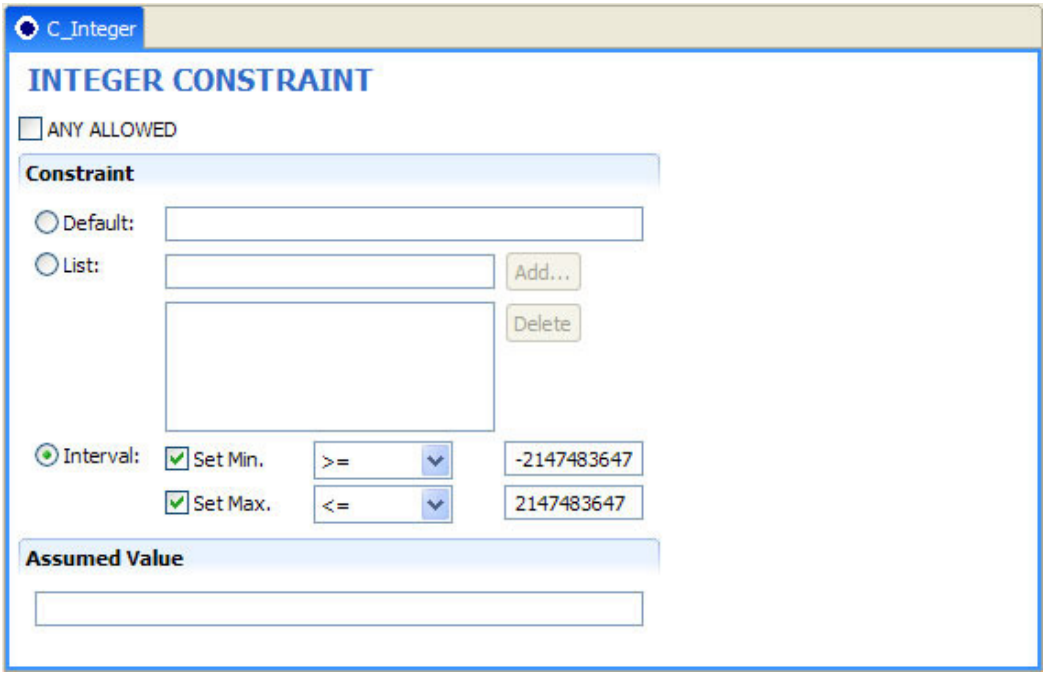
- **Archetype Slot**: Properties of an archetype slot (includes and excludes) can be edited from this form.



Capture 19: Archetype slot view

- **Primitive Object**: Each kind of primitive objects (corresponding to the basic types string, integer, double, boolean, date, time and datetime) has its own form in order to define their constraints or assumed values.





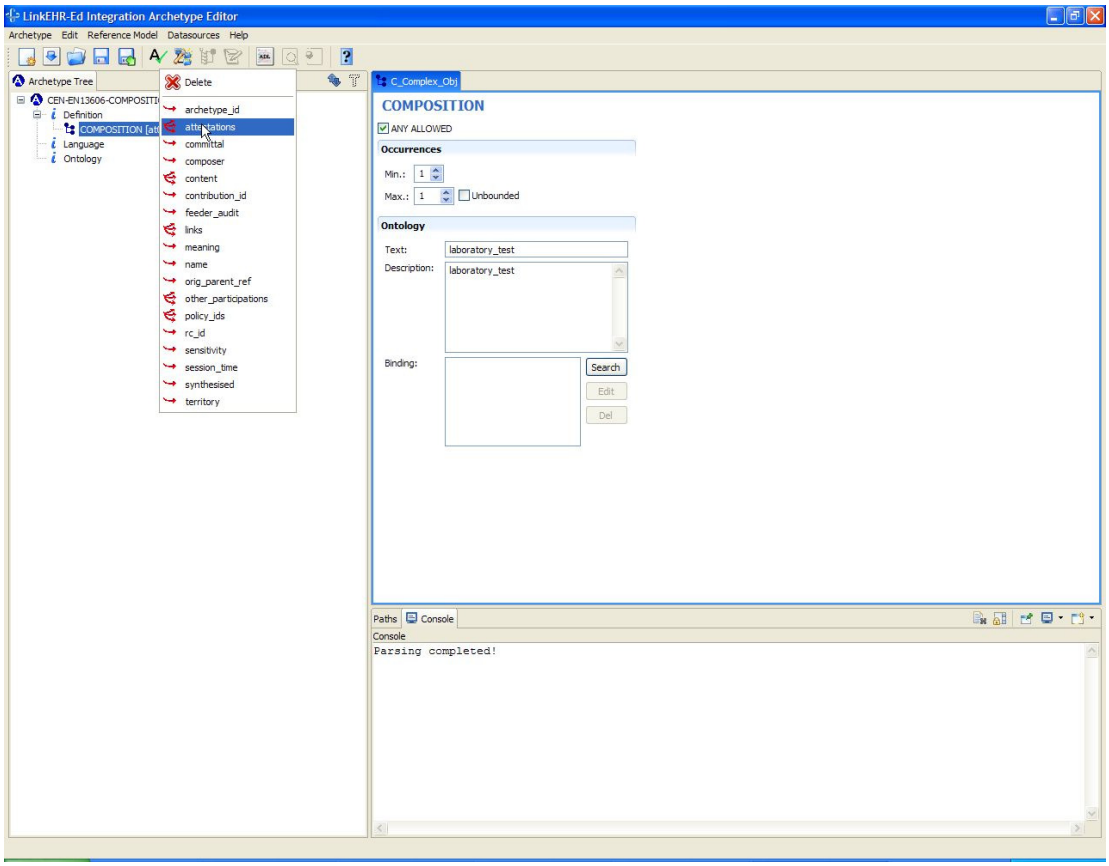
Capture 20: Samples of String and Integer primitive objects views

- **Domain Types**: Since LinkEHR-Ed is a model-independent editor, it is difficult to design an interface for editing particular domain types. Due to this fact, at this stage there is no visual interface for this kind of objects, but they can still be edited textually by switching to the ADL view.

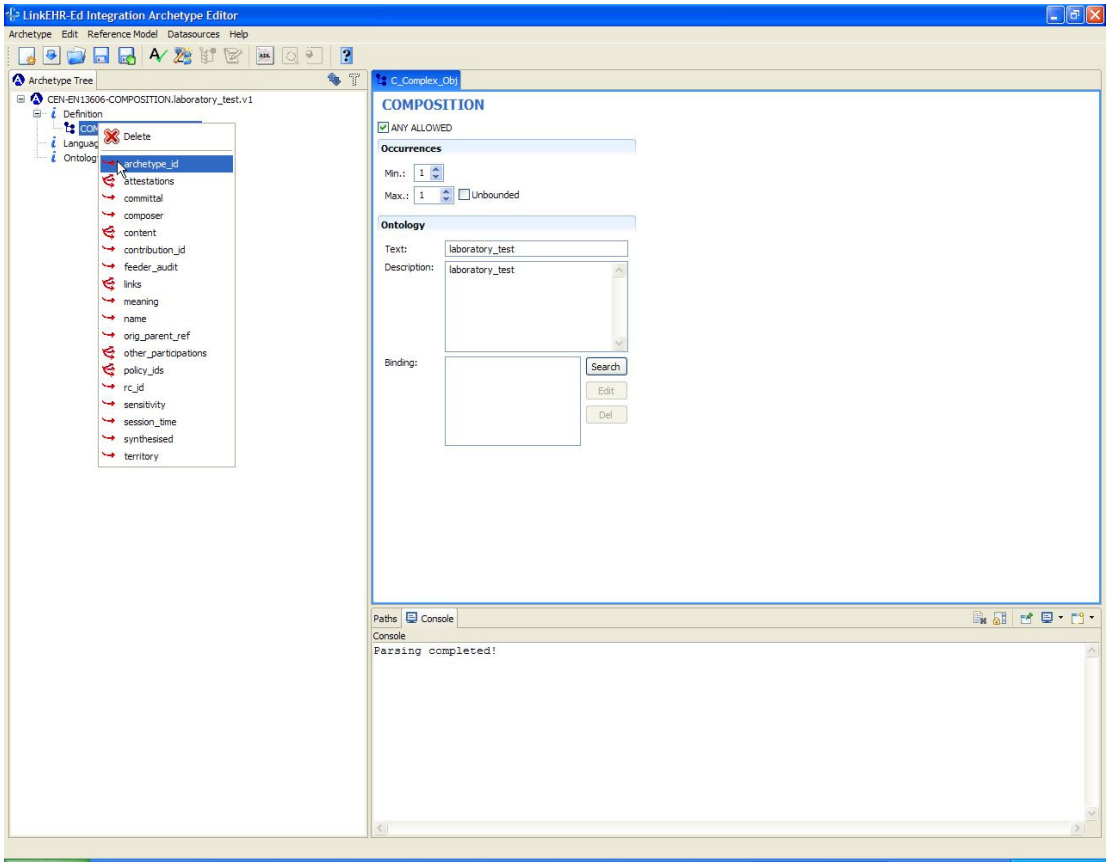
In order to edit the archetype you can add or delete new objects and attributes by clicking on the "Add archetype constraint" button of the toolbar () or by right-clicking on the node of the tree where do you want to perform those actions (captures 21 and 22). When you choose to include a new Complex Object a pop up will show up to choose which kind of object you want to add. As shown in capture 23, you can choose the new node to be an object, an internal reference to another object of the same type or an archetype slot.

You can delete a node (objects and attributes) selecting the *Delete* option from the right-click menu over the corresponding node.

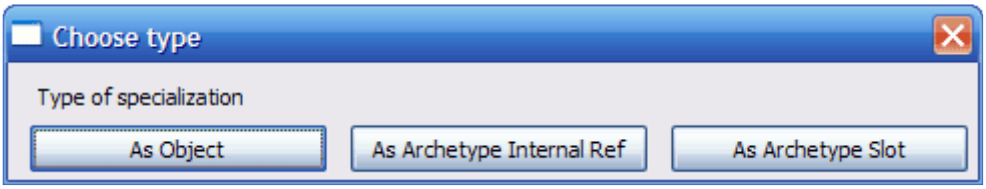
***NOTE**: When two or more objects are assigned to a Single Attribute, a virtual node called Alternatives will be created in the tree in order to show this specific fact. It is only an informative node and has no other properties.*



Capture 21: Pushing the "Add archetype constraint" button shows a menu showing the available actions…



Capture 22: …and so does clicking with right button on a node



Capture 23: Choosing the type of the node

8. Specializing an archetype

To create a specialization archetype you only have to push the "Specialize Archetype" button of the toolbar or the menu *Archetype* → *Specialize*. A dialog will be displayed to select the parent archetype of the specialization and to introduce the specialization name. Pushing Ok will create the specialization archetype, with the structure of the parent archetype already included. Then you can modify the specialized archetype the same way as told at step 7.

9. Validating an archetype

Once you have created an archetype from scratch or opened it from an available ADL file, you can validate it by pushing the Validate button  . This validation assures that the introduced constraints are effectively more constrained than those defined by a parent archetype (if this is an specialization archetype) or by the reference model. In any other case, an error message will be displayed with the path of the node where the validation failed.

10. Save an archetype

Once you have finished the edition of an archetype, you can save it trough the Save button or the *Archetype* → *Save* menu. You can choose an ADL format or an XML one (just a prototype functionality in the second case).

11. LinkEHR-Ed current limitations

LinkEHR-Ed is a prototype tool from a research project in continuous evolution. At this stage of the project it is not intended to be used in production or real environments but as a tool which gives a formal approach to the archetype edition process. Some of its current limitations are:

- Some XML Schemas of other reference models can fail, since their structure can be diverse. Currently, many of the characteristic of W3C XML schemas are supported, such as data types, name spaces, imports and includes (reference models can be defined by using several files) and a wide range of structures such as complex and simple types, elements, attributes, inheritance by extension and restriction, sequence, choice, all, attributes, patterns and groups and their respective facets.
- Domain types are not fully supported since they depend on particular reference model knowledge. They are usually correctly parsed and edited in the ADL view, but they are not shown visually.
- Access to terminology services is not yet implemented, but terminology bindings can be defined.
- Only ADL 1.4 is supported.
- Uncontrolled errors can occur. All notifications of bugs are welcome.