ELE·ON

Software and Knowledge Engineering Laboratory
Institute of Informatics and Telecommunications
National Centre for Scientific Research (NCSR) "Demokritos"
Athens, Greece.
http://www.iit.demokritos.gr/~eleon/

TABLE OF CONTENTS

OVERVIEW OF THE ELEON FUNCTIONALITY

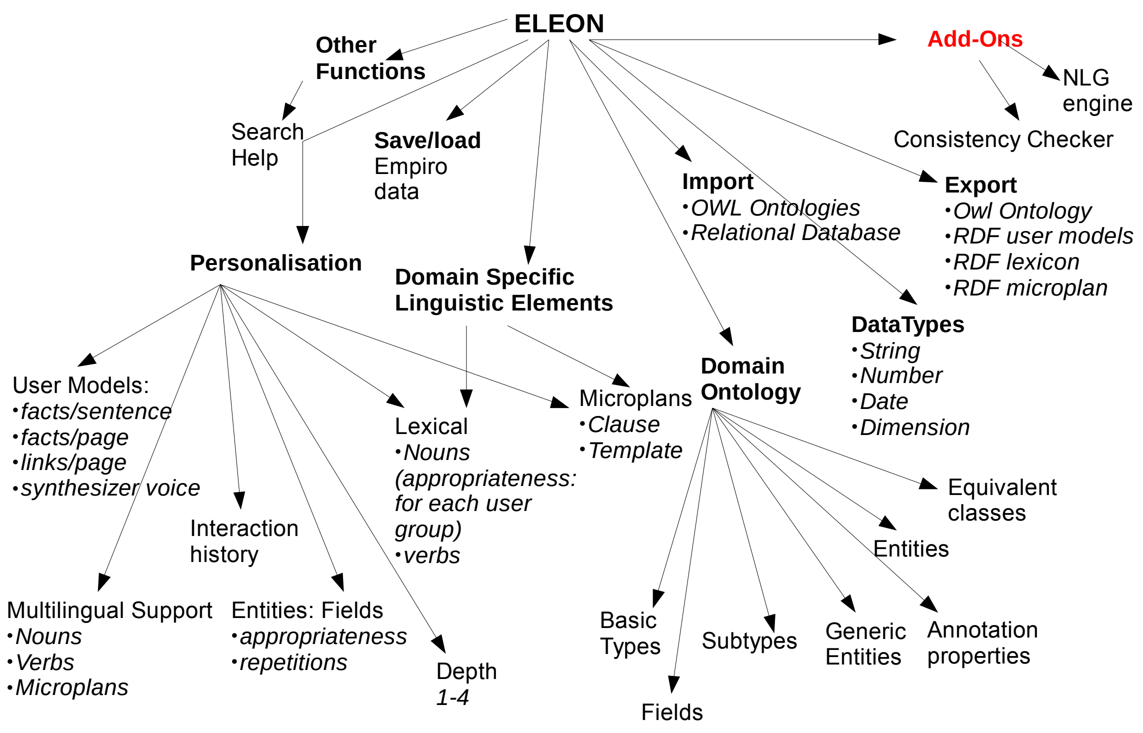


Figure 1 The ELEON authoring tool

INTRODUCTION

ELEON is, an editor that allows the enrichment of OWL ontologies with linguistic and user-related annotations. The enriched ontologies are used by natural language generation (NLG) engines to generate textual descriptions of the objects represented in the ontologies in the selected language and according to user's model. ELEON provides a well-defined interface that can be used by different NLG engines. The paper presents the relevant functionalities of ELEON, describes the provided interface to NLG engines and discusses the advantages of exploiting such enriched ontologies in NLG. (The reader is refereed to [1] for a presentation of ELEON).

INSTALLATION

Unzip the file `eleon.rar` to a directory of your choice. Under that directory there should now be the following directories and files:

`\Eleon` (the main directory)
`\Eleon\ mpiro_authoring_v4_4\mpiro_authoring_v4_4.jar` (the authoring tool .jar file)
`\Eleon\start_authoring_owl_Win.bat` (the file that starts the tool under Windows)
`\Eleon\libs` (library files)

You are also going to need a natural language generation engine (NLG) to preview the text that can be generated. Download the NaturalOWL, from the Natural Language Processing Group of the Department of Informatics, Athens University of Economics and Business[1]. The file should be named `NaturalOWL.tar.gz`. Unzip the file, and detect the file `NL.jar.` This is the file that you should copy to the directory `libs`. Also, the Java runtime engine will be necessary. In addition ir

[1] http://pages.cs.aueb.gr/nlp/software.html
http://www.racer-systems.com/

DOMAIN ONTOLOGY

Basic Types

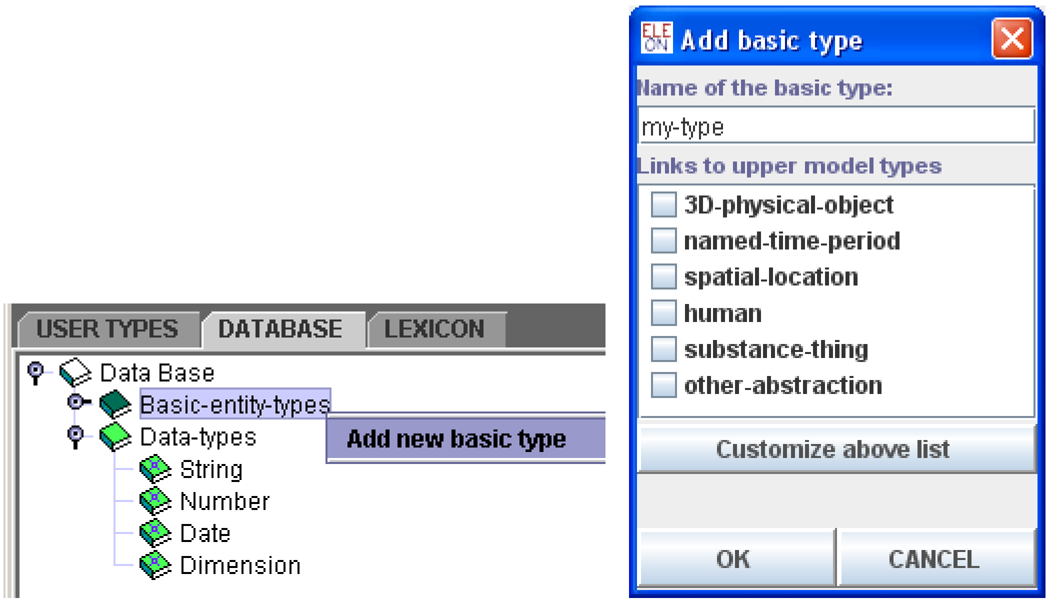
Description:
Basic types are classes that may contain other classes but only of *subtype* genre and not of the *basic type* genre or *instances*.

Declaring basic types
The author must declare at least one basic type. There are two characteristics of a basic type: First, it exists at the top level of the hierarchically organized ontology. Second, it must have a link to pre-existing *upper model type*. The upper model types are: *3D-physical-object, named-time-period, spatial-location, human, substance-thing, other-abstraction*. Should the upper models prove to be inadequate; the author can specify its own *upper model type*. Finally, the author specifies the name of the basic type. Each basic type must have a unique name in the ontology.

Editing basic types
The basic types can be deleted or their name can be edited. In addition another type can be added as subtype of the current basic type. Each basic type has some predefined *fields* and users can add more fields.

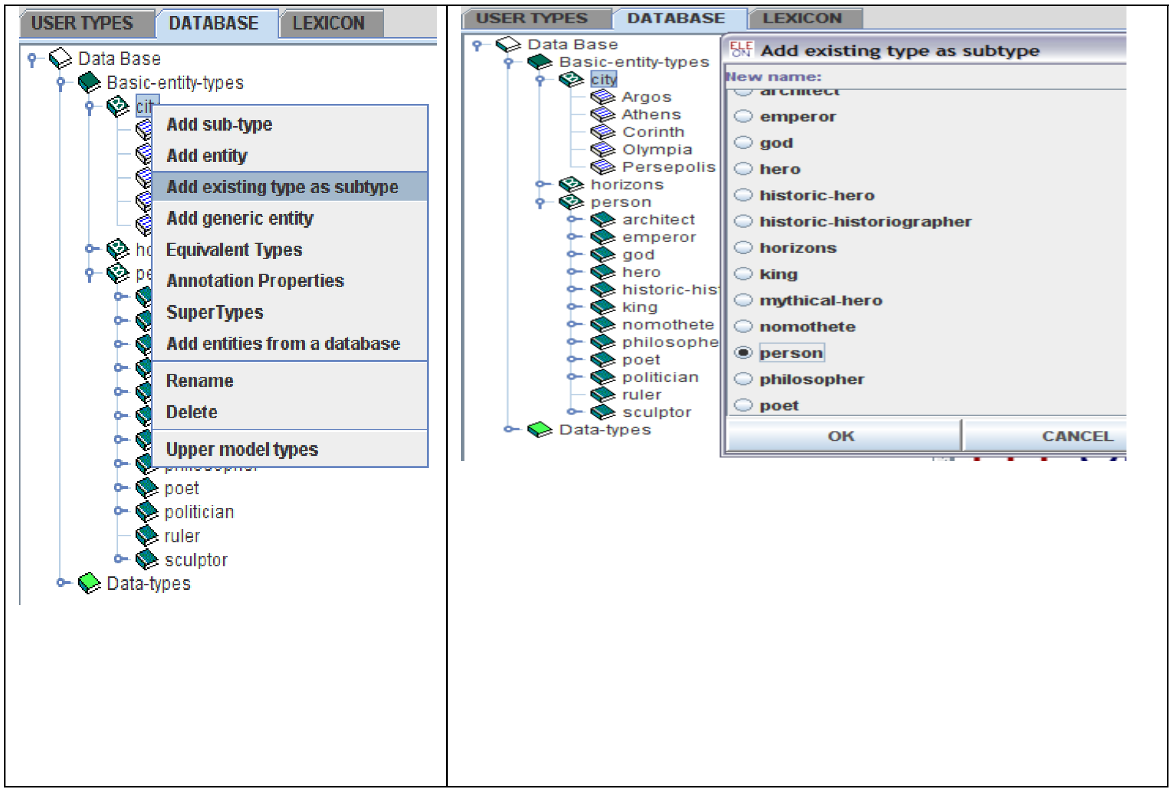
Basic types and Hierarchy
Each basic type can be added as subtype of one or more subtypes

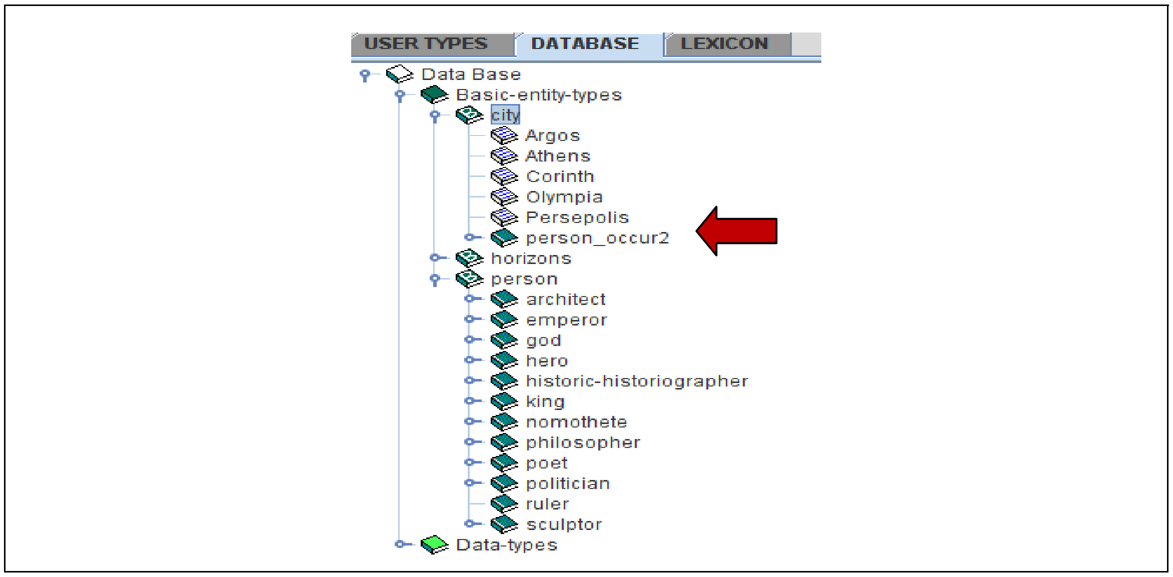
Examples:



Name of the basic type: my-type

Figure 2 Creating a basic type





Data Base
Basic-entity-types
city
Argos
Athens
Corinth
Olympia
Persepolis
person_occur2
horizons
person
architect
emperor
god
hero
historic-historiographer
king
nomothete
philosopher
poet
politician
ruler
sculptor
Data-types

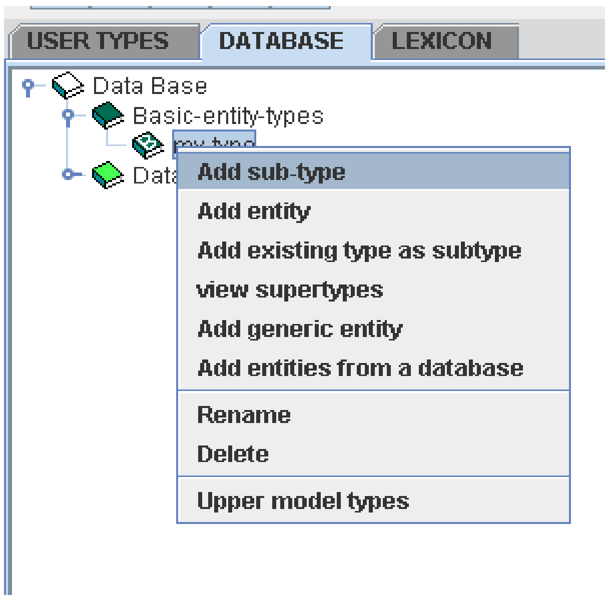
Figure 3Adding the "person" basic type as subtype of "city". At the top left, we select the basic type to which we shall add an existing type as subtype. At the top right, we tell ELEON to add person as a subtype of "city". At the bottom we see under city, a copy of person.

Sub Types

Description:
Subtypes are classes that may contain other classes but only of *subtype* genre and not of the *basic type* genre or *instances*. Subtypes may not exist at the top level of the ontology and they do not require an upper model type.

Properties:
Each basic type has some predefined *fields* and users can add more fields. The fields serve the role of describing a subtype.

Examples:



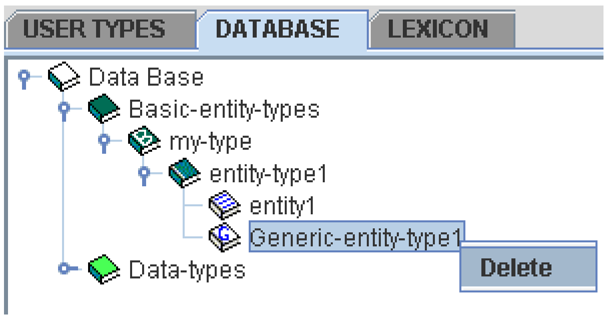
Generic Entities

Description:
A *Generic Entity* is an instance of a *basic type* or a *subtype.* Each basic type or subtype can have only one generic entity. The name of the generic entity is derived from the type or subtype in which it belongs to. Generic entities inherit the properties of the type (or subtype) they belong to. Each generic entity field can assume multiple values.

Entities

Description:
Entities are instances of *basic types* or *subtypes.* Entities inherit the properties of the type (or subtype) they belong to. Each basic type or subtype can have multiple entities.

Examples in ELEON:



Fields

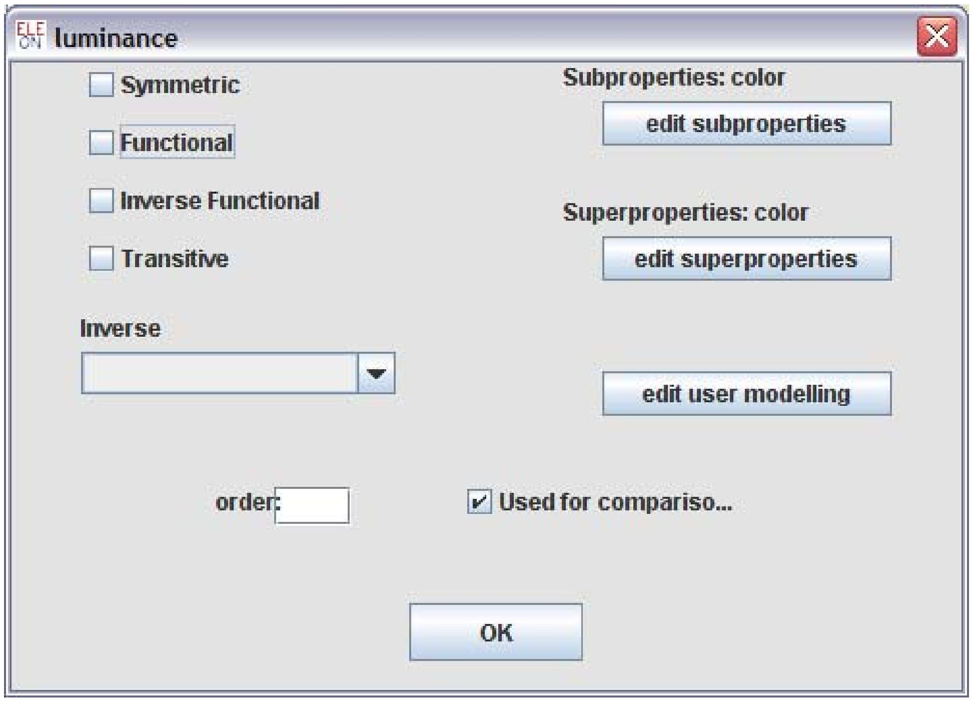
Description: *Fields* denote properties in types, subtypes, generic entities and entities. There are prespecified fields and fields that are user defined. The prespecified fields are *subtype-of, title, name, shortname, notes, gender, number* and *images.*

Fields declared in types are inherited by subtypes and entities (or generic entities). Generally, speaking fields are inherited from higher items of the hierarchy to lower items. Moreover, fields can be added to types and subtypes only. The type of the user defined fields (appearing as filler-types) must be declared. There is some predefined filler-types: the string, date, dimension, and number. The rest of the possible filler types are derived from the types and subtypes that have been defined.

For every field that belongs to a type or a subtype: *properties*, *restrictions*, *importance* and *repetitions* can be defined.

Properties

- Symmetric: when a property is symmetric, if an entity A is related through this property with another entity B, then, in return, the entity B is related through the same property with entity A. For example, if the entity "building1" is related through the property "is-next-to" to the entity "building2" and the property "is-next-to" is symmetric, then "building2" is related to "building1" by the "is-next-to" property.
- Functional: a property is functional if every entity has only one value for this property.
- Inverse Functional: a property is inverse functional if the inverse (see below) of this property is functional. If a property P1 has another property P2 as inverse property and entity A is related to entity B by P1, then B is related to A by P2
- Transitive: if a property is transitive and an entity A is related to another entity B by this property and also B is related to a third entity C, then A is also related to C by this property.
- Also for every field the author can set the *subroperties* and *superproperties* for this field, creating thereby a property hierarchy. Thus, for example if a property P1 is sub-property of another property P2 and entity A is related to entity B by P2, then A is also related to B by P1. The *order* refers to whether the natural language engine will talk about the value of the filler. A value of zero denotes that this feature is not active, one instructs the engine will produce text about the filler. A value of two, instructs the engine to talk about the filler's filler and so on.
- The *edit user modeling* allows the author to specify the default interests and repetitions for this field. Finally, the author can set whether this field will be used for *comparisons* by a natural language generation engine.



Restrictions
The following restrictions to the values of a field (of a basic type or a subtype) can be defined: *all values from, some values from, has values, min cardinality* and *max cardinality*. The 'values' predicate refers to values from a basic type or a subtype.

Importance & Repetitions
The importance of the field can be denoted for each of the user model. For instance, the importance of a certain field might differ between adults and children. Finally, repetitions refer to the maximum number of times a field can be used, before it is consider as assimilated information by the user. Importance and repetitions, affect the text produced by the NLG system.

Annotation Properties

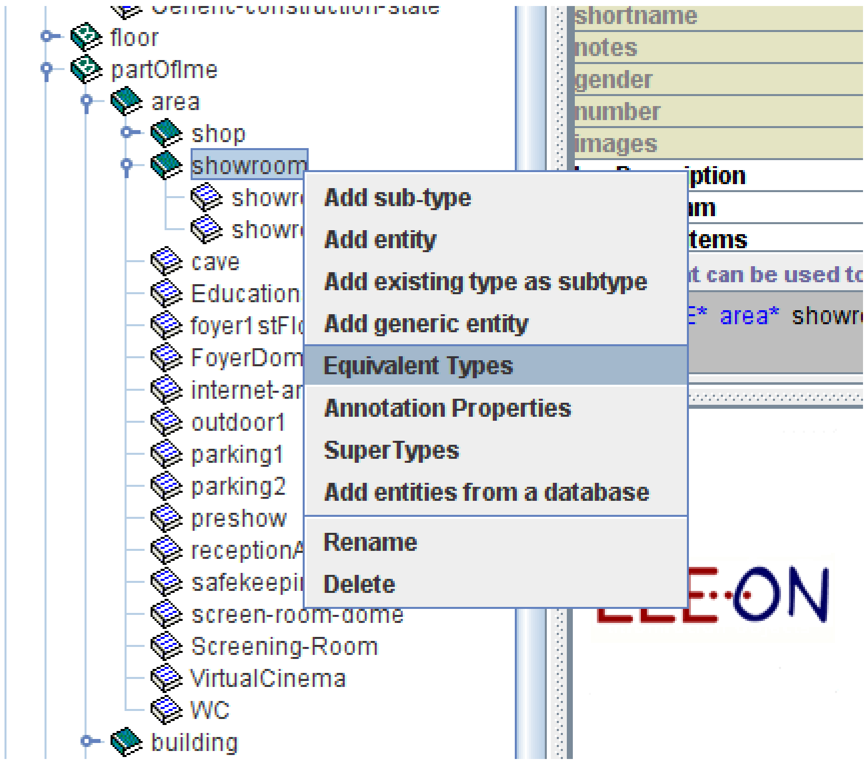
Description
It supports the five OWL predefined annotation properties: *owl:versionInfo, rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefinedBy*

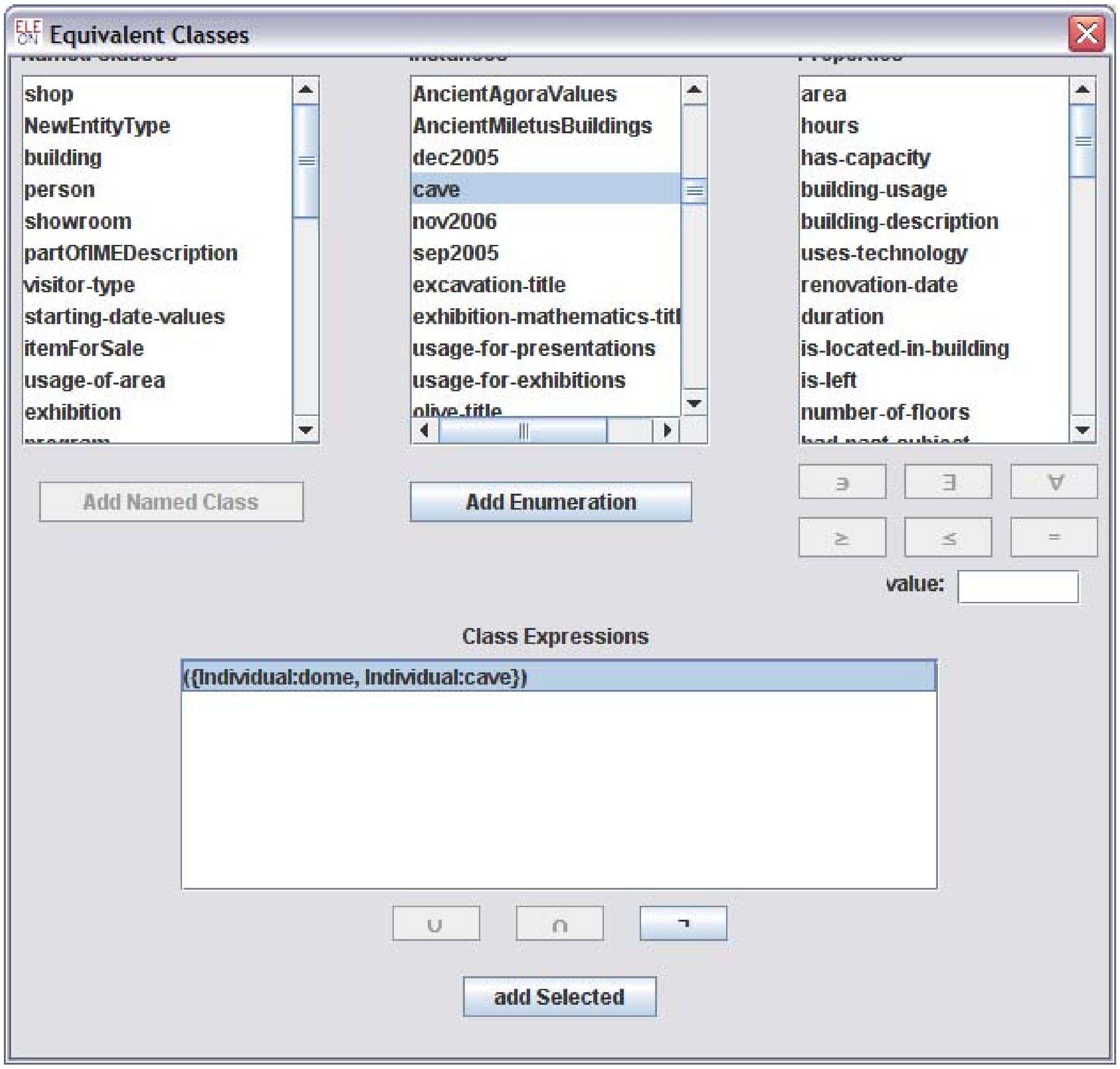
Examples

Equivalent Classes

Description:
The author can define equivalence relations between classes (i.e. basic types or subtypes), in the sense of linking a class description to another class description. In addition the author can create a class and an enumeration

Examples:
The first column records the classes, the second the entities and the third the properties. Let us assume that the author wishes to say that the class *showroom* is equivalent the *dome* and *cave* entities.





({Individual:dome, Individual:cave})

The way to define an equivalence relation is through the *Add Named Class* or the *Add Enumeration.*

DOMAIN SPECIFIC LINGUISTIC DATA

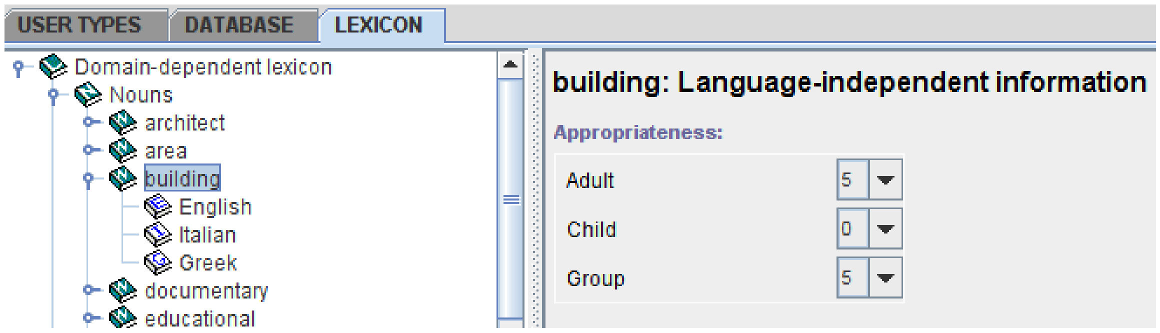
Lexical

Authors using the ELEON can record lexical information in the form of nouns and verbs for English, Greek and Italian that form the domain specific dictionary.

NAME: Noun

DESCRIPTION:
The point of defining nouns, is to use them in the expression of types during the natural language generation process. In addition, authors can specify the degree of *appropriateness* of each noun for each user model.
Authors, can add a new *noun* by providing an identity name. Then the author has to specify the forms the noun assumes in various languages (English, Italian and Greek), which depend on the idiosyncrasy of each language. For instance, the singular and plural form across cases can be specified, in addition to the gender and whether it is countable or uncountable. Finally, the authors can remove nouns.

EXAMPLES:
Defining the appropriateness of the noun building for each user model: Adult, Child and Group on a scale of -5 to 5. The smaller the number the less appropriate it is.

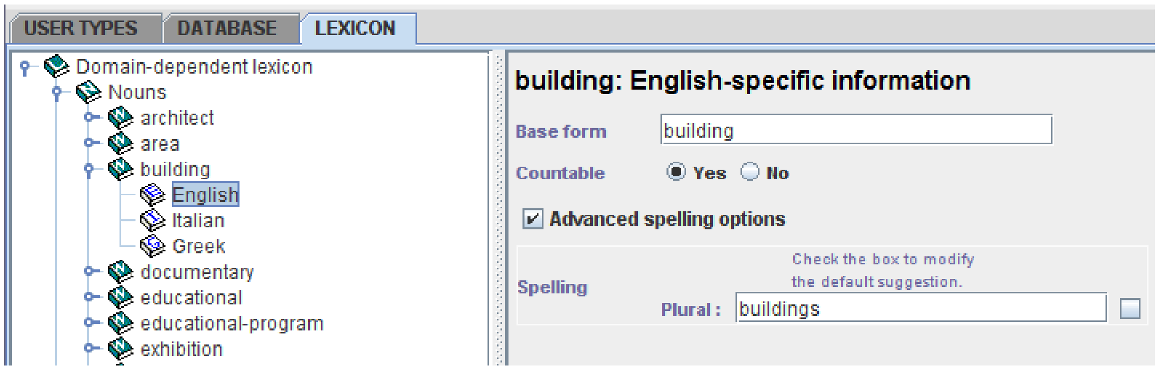


building: Language-independent information

Appropriateness:

Adult	5
Child	0
Group	5

Defining different aspects of the building noun for English



building: English-specific information

Base form: building

Countable: [✓ Yes] No

[✓] Advanced spelling options

Spelling — Check the box to modify the default suggestion.

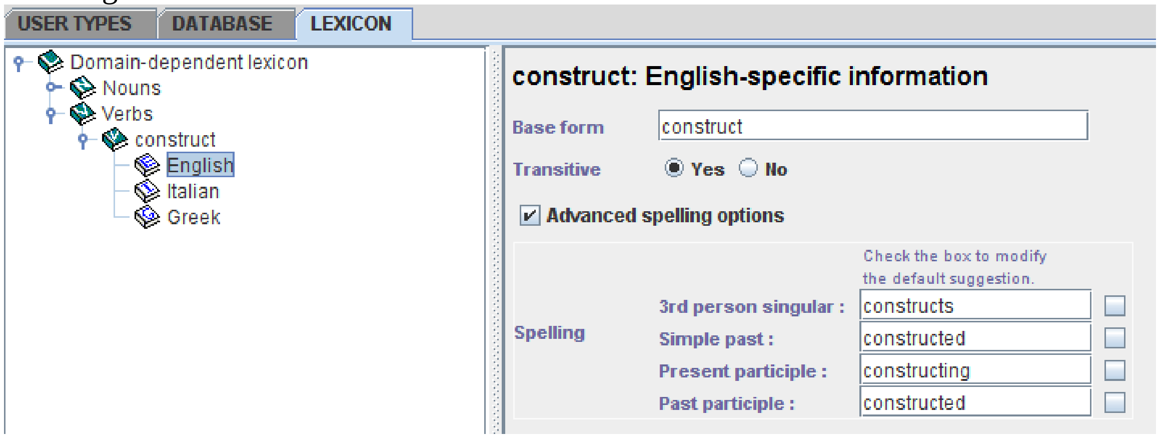
Plural : buildings

NAME: Verb

DESCRIPTION:

EXAMPLES:

Defining a verb



construct: English-specific information

Base form: construct

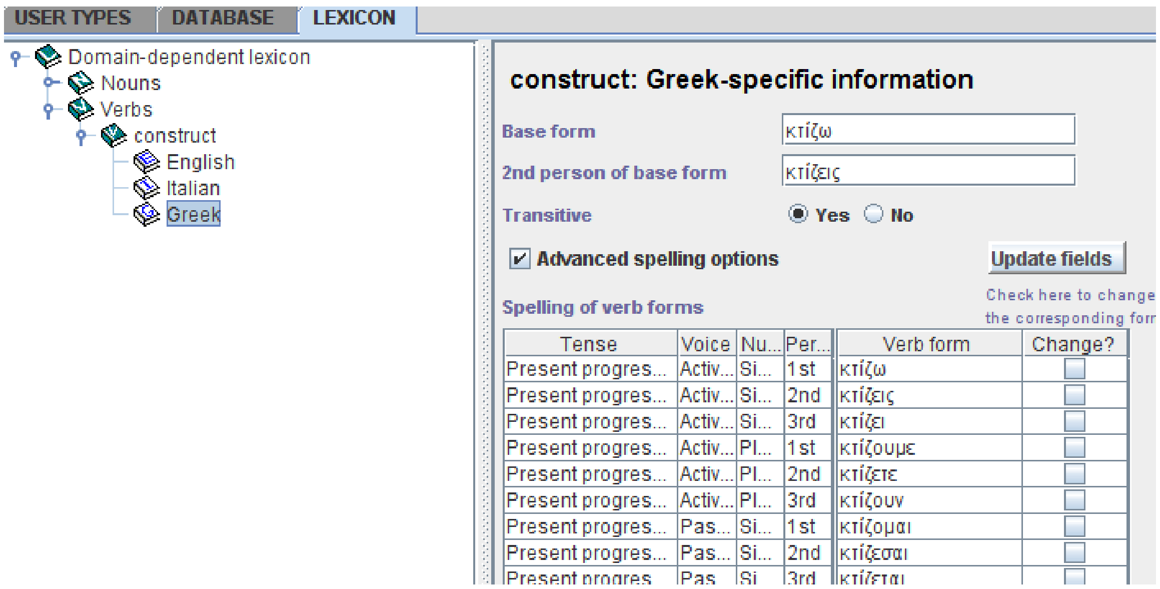
Transitive: [✓ Yes] No

[✓] Advanced spelling options

Spelling — Check the box to modify the default suggestion.

3rd person singular :	constructs
Simple past :	constructed
Present participle :	constructing
Past participle :	constructed

For inflected languages (in particular in Greek and in Italian), there are more options



construct: Greek-specific information

Base form	κτίζω
2nd person of base form	κτίζεις
Transitive	[✓ Yes] No

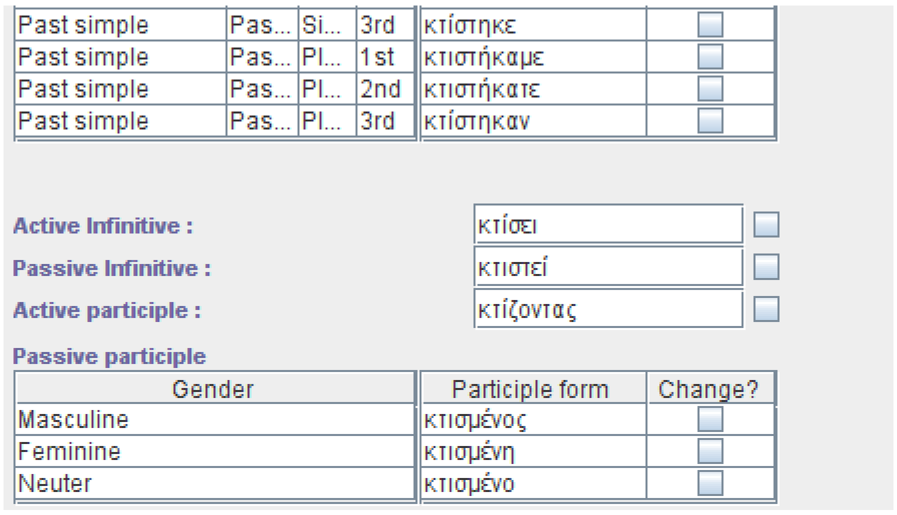
[✓] Advanced spelling options

Spelling of verb forms

Check here to change the corresponding forr

Tense	Voice	Nu...	Per...	Verb form	Change?
Present progres...	Activ...	Si...	1st	κτίζω	☐
Present progres...	Activ...	Si...	2nd	κτίζεις	☐
Present progres...	Activ...	Si...	3rd	κτίζει	☐
Present progres...	Activ...	Pl...	1st	κτίζουμε	☐
Present progres...	Activ...	Pl...	2nd	κτίζετε	☐
Present progres...	Activ...	Pl...	3rd	κτίζουν	☐
Present progres...	Pas...	Si...	1st	κτίζομαι	☐
Present progres...	Pas...	Si...	2nd	κτίζεσαι	☐
Present progres	Pas	Si	3rd	κτίζεται	☐

……………………………………..



Past simple	Pas...	Si...	3rd	κτίστηκε	☐
Past simple	Pas...	Pl...	1st	κτιστήκαμε	☐
Past simple	Pas...	Pl...	2nd	κτιστήκατε	☐
Past simple	Pas...	Pl...	3rd	κτίστηκαν	☐

Active Infinitive :	κτίσει	☐
Passive Infinitive :	κτιστεί	☐
Active participle :	κτίζοντας	☐

Passive participle

Gender	Participle form	Change?
Masculine	κτισμένος	☐
Feminine	κτισμένη	☐
Neuter	κτισμένο	☐

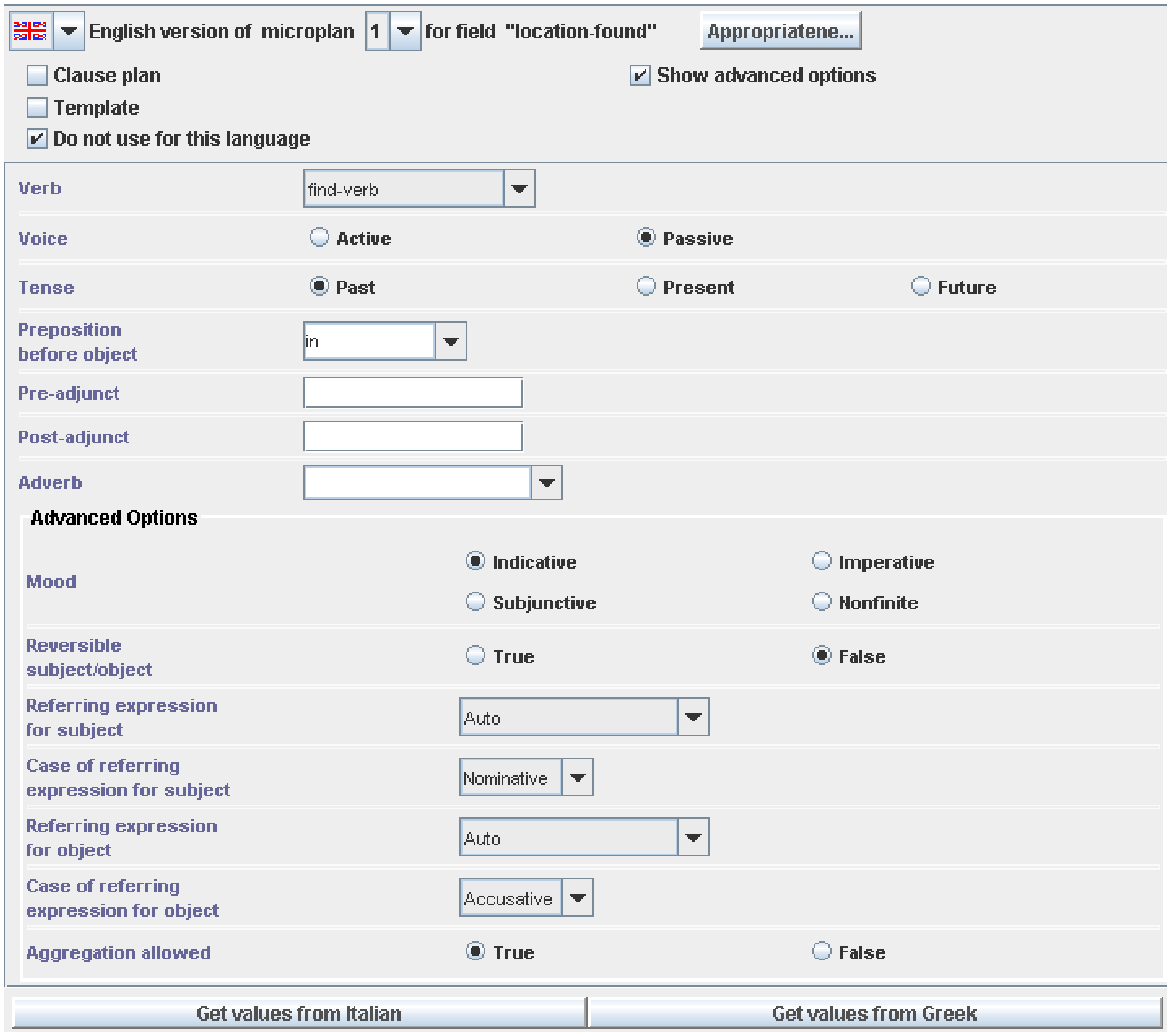
Microplans

The ELEON tool offers two ways of entering linguistic information about the text that will produced for each field of the type (class). In particular there the clause microplan and the template microplan (the first being more versatile than the second at the expense of being less comprehensible by non-experts). In particular, there can be up to five microplans for each field.

NAME: Clause Microplan

DESCRIPTION:

EXAMPLES:
Next follows an example of a clause microplan.


English version of microplan 1 for field "location-found"

Appropriatene...

- ☐ Clause plan
- ☐ Template
- ☑ Do not use for this language
- ☑ Show advanced options

Verb	find-verb		
Voice	Active	[✓ Passive]	
Tense	[✓ Past]	Present	Future
Preposition before object	in		
Pre-adjunct			
Post-adjunct			
Adverb			

Advanced Options

Mood	[✓ Indicative]	Imperative
	Subjunctive	Nonfinite
Reversible subject/object	True	[✓ False]
Referring expression for subject	Auto	
Case of referring expression for subject	Nominative	
Referring expression for object	Auto	
Case of referring expression for object	Accusative	
Aggregation allowed	[✓ True]	False

Get values from Italian | Get values from Greek

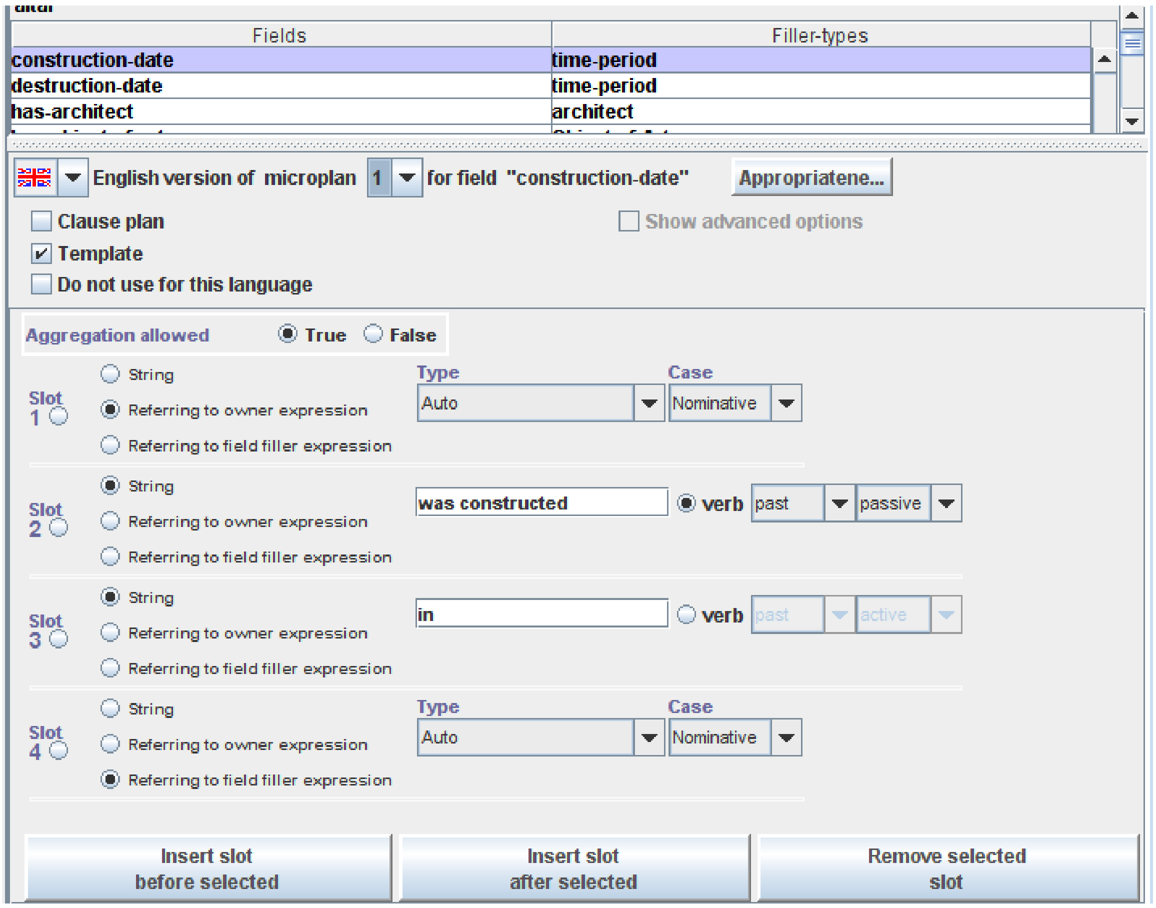
NAME: Template Microplan

DESCRIPTION:
A *template Microplan* is a rather strict way – compared to clause microplans - of describing the way a specific field (of a type or subtype) field will be expressed by a natural language generation engine. A template is made of a set of successive slots that contain the linguistic information. Each slot can be of three types: *a string*, *a reference to owner expression* or a *field filler*. In the string case, the author specifies a string, and he might also denote whether it is verb, its tense and voice; this information is mostly useful for the natural generation engine. A reference to owner (of the field) will fill the slot with the name of the type or subtype in which it belongs to. The user has the option of determining the case (nominative, genitive, accusative) and whether the owner's name is a noun, pronoun, it has a definite or indefinite article; again this information is transferred

to a natural language generation engine. Furthermore the author can denote whether this microplan can be integrated with other microplans to create compound sentences instead of separate ones.

Example

The following is an example of a *template microplan* for the *construction-date* field (highlighted in blue). Four slots (parts) constitute the current microplan. The first slot is of the type *referring to owner expression* which means that the value is obtained from the corresponding entity name. The next two slots are strings, and in particular for the second slot the author of the microplan adds some information (about the tense and the voice of the verb), which might be exploited by a natural language generation engine. In the last slot, *field filler* denotes that the value stems from whatever the user enters in the construction-data field for a specific entity.



Fields	Filler-types
construction-date	time-period
destruction-date	time-period
has-architect	architect

English version of microplan 1 for field "construction-date" Appropriatene...

- [ ] Clause plan
- [✓] Template
- [ ] Do not use for this language
- [ ] Show advanced options

Aggregation allowed [✓ True] False

Slot 1: String [✓ Referring to owner expression] Referring to field filler expression — Type: Auto; Case: Nominative

Slot 2: [✓ String] Referring to owner expression Referring to field filler expression — was constructed [✓ verb] past passive

Slot 3: [✓ String] Referring to owner expression Referring to field filler expression — in verb past active

Slot 4: String Referring to owner expression [✓ Referring to field filler expression] — Type: Auto; Case: Nominative

PERSONALISATION

ELEON, as it has been mentioned is more than an ontology authoring tool, in particular it offers a substantial personalization functionality, which is realized through the following functions:

- User types
- Multilingual support for microplans, nouns and verbs
- Appropriateness of each microplan for each User Type
- Importance and Repetitions values for each field
- Interaction History of the User

DESCRIPTION: It supports functions that provide personalization to user's characteristics, such as: *mother tongue, knowledge level, interests* and *interaction history*.

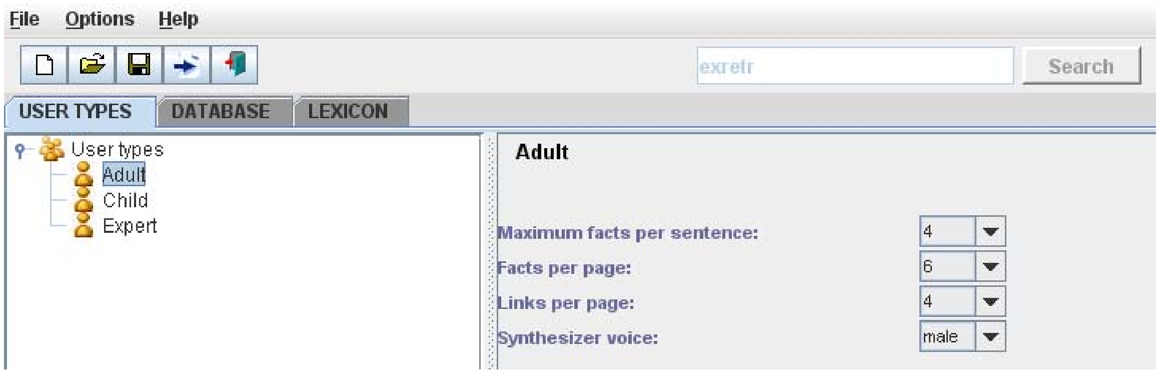
IMPLEMENTED IN ELEON:
Personalisation is implemented through trilingual support for lexical elements (nouns and verbs) and microplans. Also, it is implemented as *appropriateness* and *repetitions* for fields.

Multilingual Support

Description:
Multilingual support is realised for English, Greek and Italian. In particular language specific features can be defined for: *Nouns, Verbs* and *Microplans.*

User Types

Description:
The author can define user types, and for each user type to define the *maximum number of facts per sentence, facts per page, links per page* and *Synthesiser voice.* The number of facts refers to the number of microplans that will be employed (bear in mind that two or more microplans can be aggregate by the natural language generation engine to create a single sentence). Number of links, is no longer used. Finally, synthesiser voice is a choice that refers to the text to text speech program.

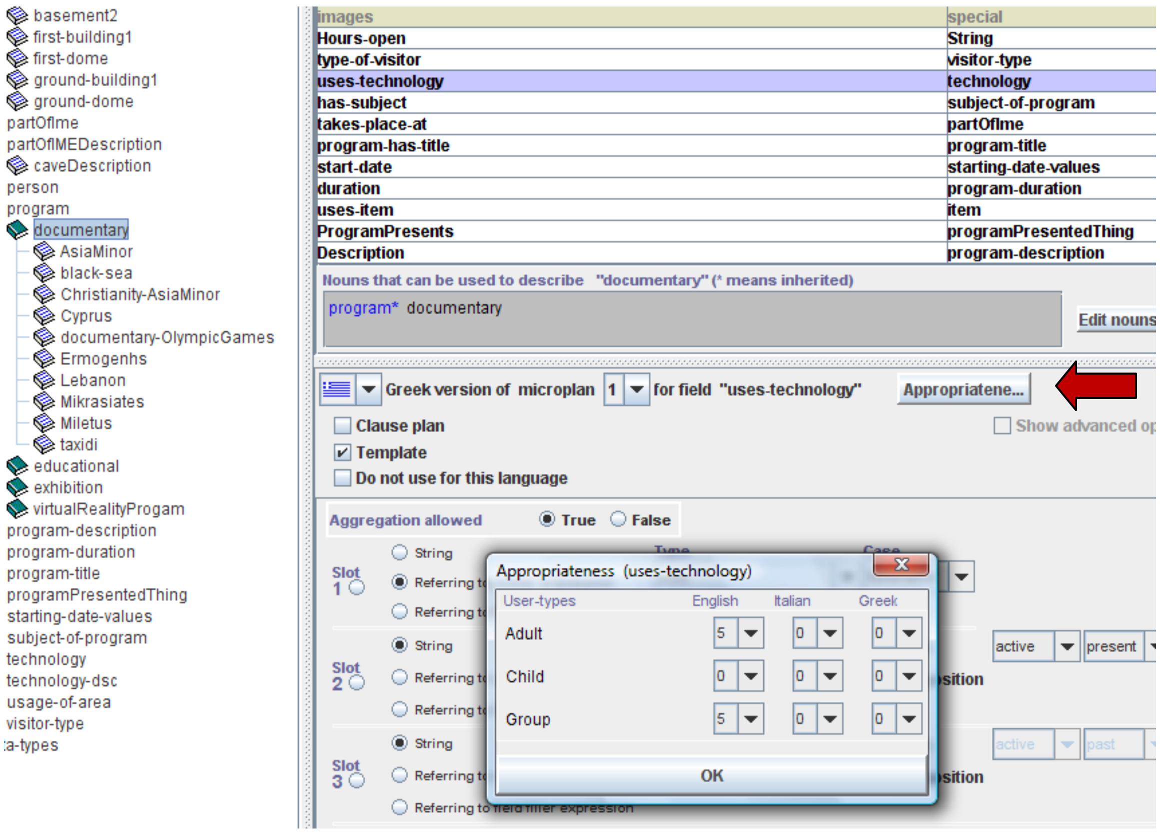


Adult

Maximum facts per sentence:	4
Facts per page:	6
Links per page:	4
Synthesizer voice:	male

Appropriateness

If multiple microplans are available for a field, then a value of appropriateness can set for each microplan, and for each user type. This is a suggestion to the NLG engine to select the "right" microplan for the current user. For instance, there might be two microplans, one set for children and the other for adults. The value of appropriateness is set as shown in the next figure:



images	special
Hours-open	String
type-of-visitor	visitor-type
uses-technology	technology
has-subject	subject-of-program
takes-place-at	partOfme
program-has-title	program-title
start-date	starting-date-values
duration	program-duration
uses-item	item
ProgramPresents	programPresentedThing
Description	program-description

Nouns that can be used to describe "documentary" (* means inherited)

program* documentary

Greek version of microplan 1 for field "uses-technology"

Clause plan [✓ Template] Do not use for this language

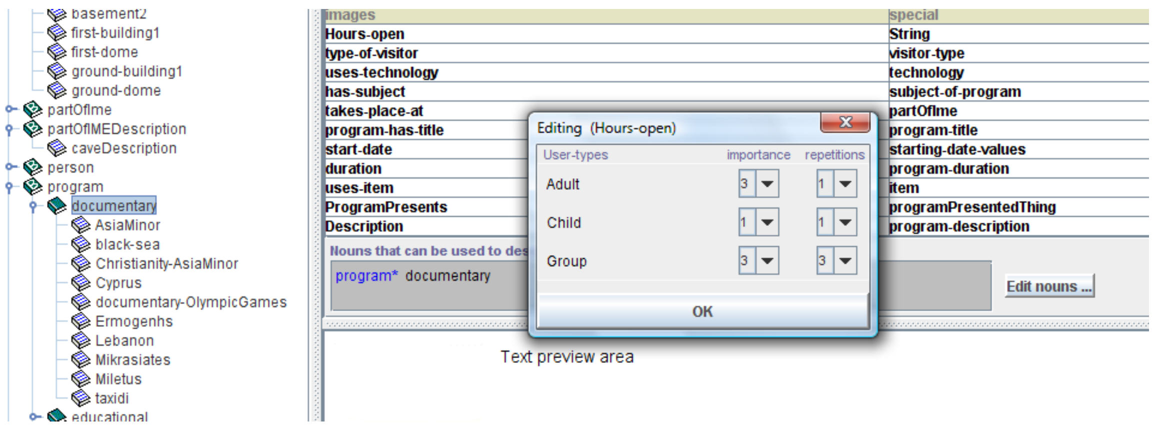
Aggregation allowed [✓ True] False

Appropriateness (uses-technology)

User-types	English	Italian	Greek
Adult	5	0	0
Child	0	0	0
Group	5	0	0

Importance

Importance refers to the importance of a field for a specific user type. Low importance for, is a suggestion to the NLG engine not to produce text for this field. For instance, in the next figure the importance of the field Hours-open, has been to 3 for the adults and to 1 for children. This can be set by right clicking on the appropriate field.



basement2
first-building1
first-dome
ground-building1
ground-dome
partOfme
partOfMEDescription
caveDescription
person
program
documentary
AsiaMinor
black-sea
Christianity-AsiaMinor
Cyprus
documentary-OlympicGames
Ermogenhs
Lebanon
Mikrasiates
Miletus
taxidi
educational

images	special
Hours-open	String
type-of-visitor	visitor-type
uses-technology	technology
has-subject	subject-of-program
takes-place-at	partOfme
program-has-title	program-title
start-date	starting-date-values
duration	program-duration
uses-item	item
ProgramPresents	programPresentedThing
Description	program-description

Nouns that can be used to des
program* documentary

Editing (Hours-open)

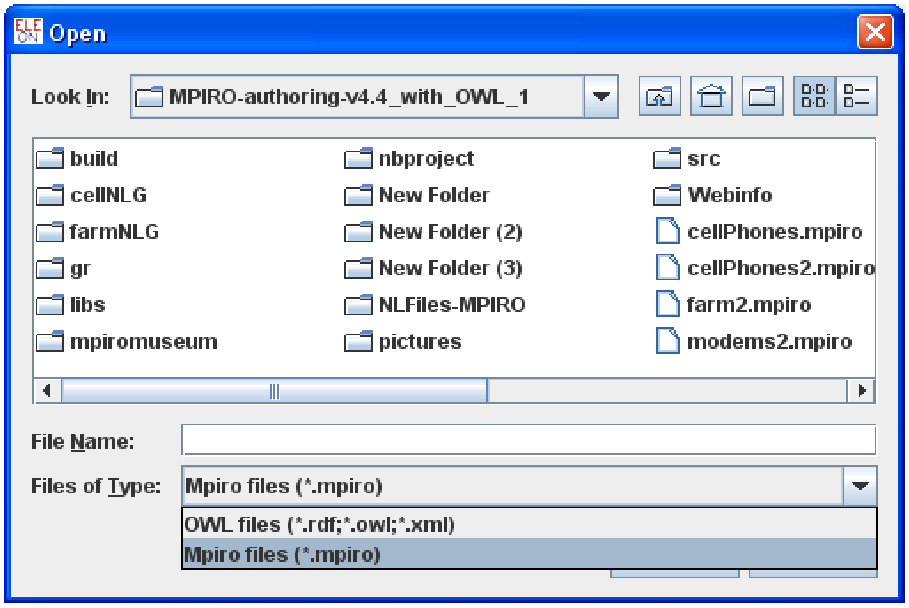
User-types	importance	repetitions
Adult	3	1
Child	1	1
Group	3	3

Interaction History

Interaction history is some sort of log file that contains the exhibits a user has visited, as well as the number of time he has visited an exhibit. The interaction history also defines how many times text can be produced for a field, before the system considers the information as being assimilated. This is set in the repetitions column (see the above figure).

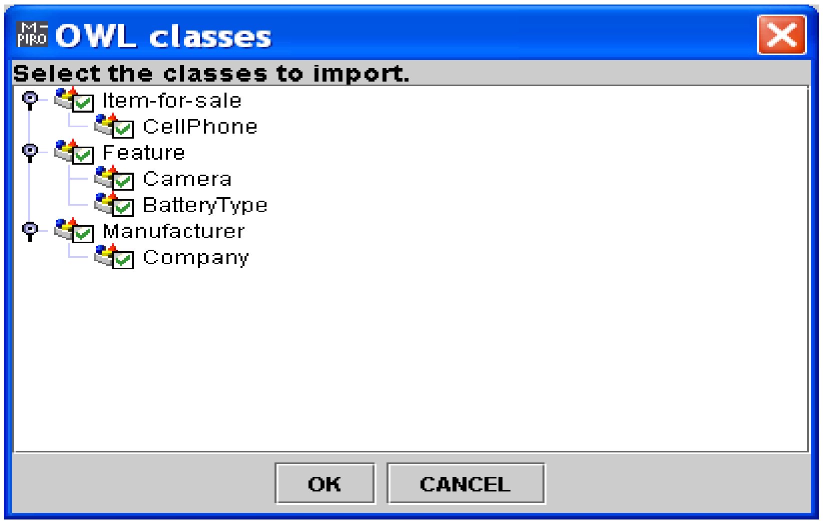
IMPORT

DESCRIPTION:
The ELEON authoring tool can import ontologies in OWL-DL form from a disk file. It can also import .mpriro (its native) format files, which can incorporate Ontologies, User models, microplans and a lexicon. In addition it can import Ontologies from relational databases.



Importing OWL Ontologies

DESCRIPTION:
It can import files in OWL-DL format. Such a file will contain the ontology of a domain. The author has the ability to choose the classes and instances that will be imported.



Importing Mpiro Files

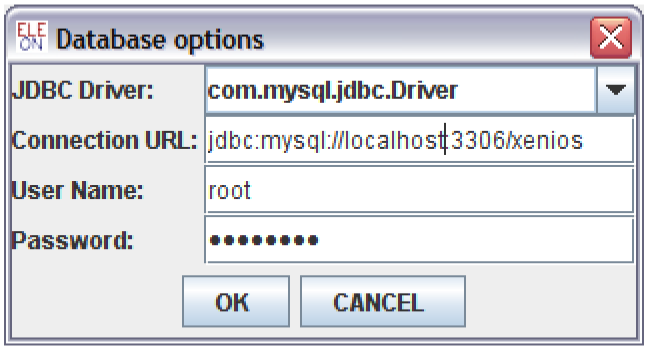
DESCRIPTION:
It reads an mpiro format file (it is a binary file, representing a Java Object), which potentially contains an ontology (types and entities), user models, microplans, a lexicon.

Importing Relational Databases

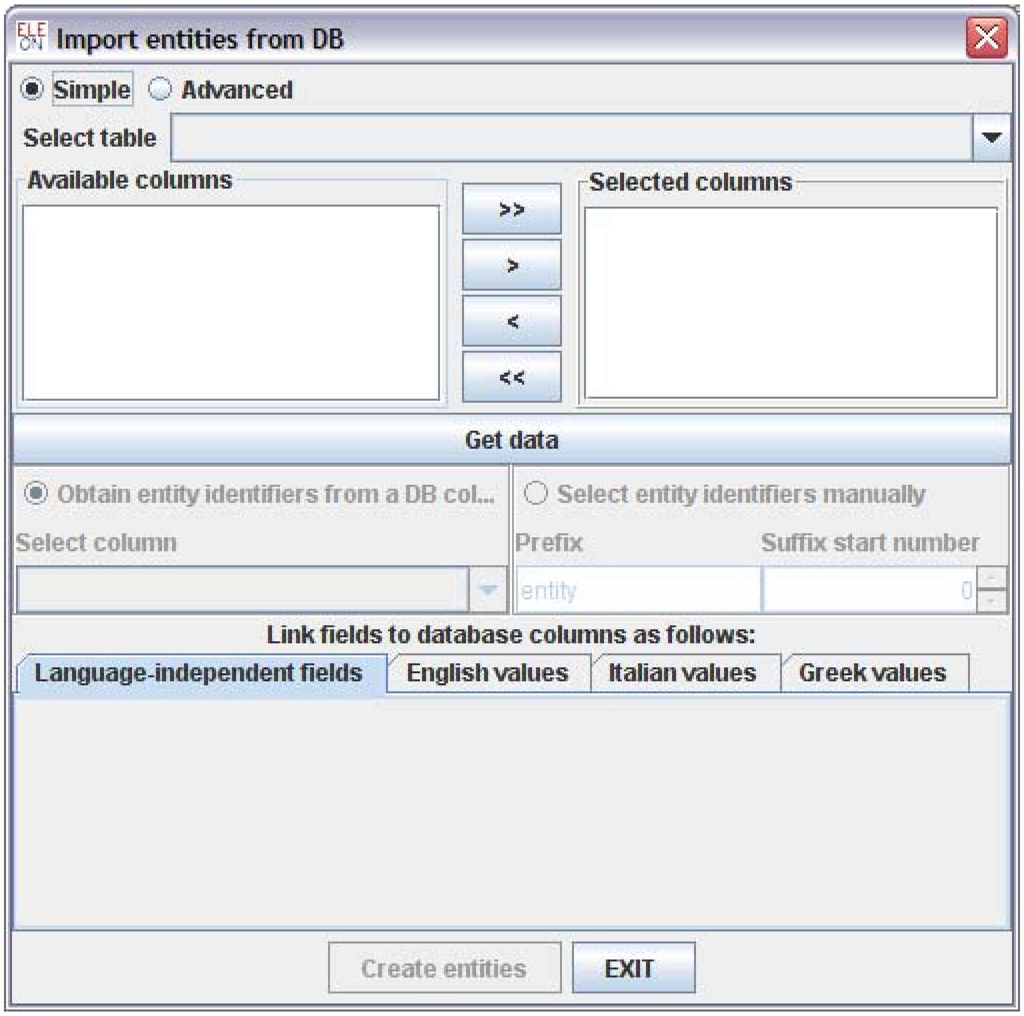
Entities can be imported from a relational database. The entities will be incorporated on a specific basic type, or subtype.

Example

To connect to a mysql database you should enter the following information. In the first field, it is the JDBC driver, in the second field it is the URL of the database (in the current example it resides on the same computer on port 3306) and the string after the last slash is the database name



JDBC Driver: com.mysql.jdbc.Driver
Connection URL: jdbc:mysql://localhost:3306/xenios
User Name: root
Password: ●●●●●●●●●



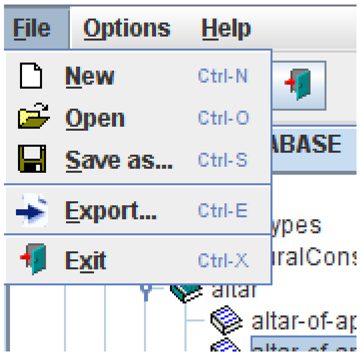
Prefix: entity
Suffix start number: 0

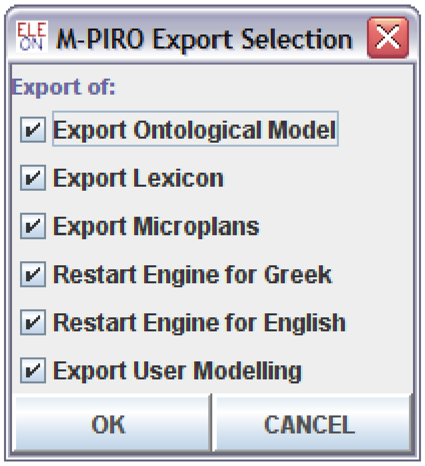
EXPORT

The enriched domain ontology which comprises the OWL ontology, the user models, the lexicon and the microplans can be exported into four files which are as follows:

- The ontology is exported as an OWL file: OwlTemp.owl
- The user models an RDF file: UserModelling.rdf
- The lexicon as an RDF file: Lexicon.rdf
- The microplans as an RDF file: Microplans.rdf

Example





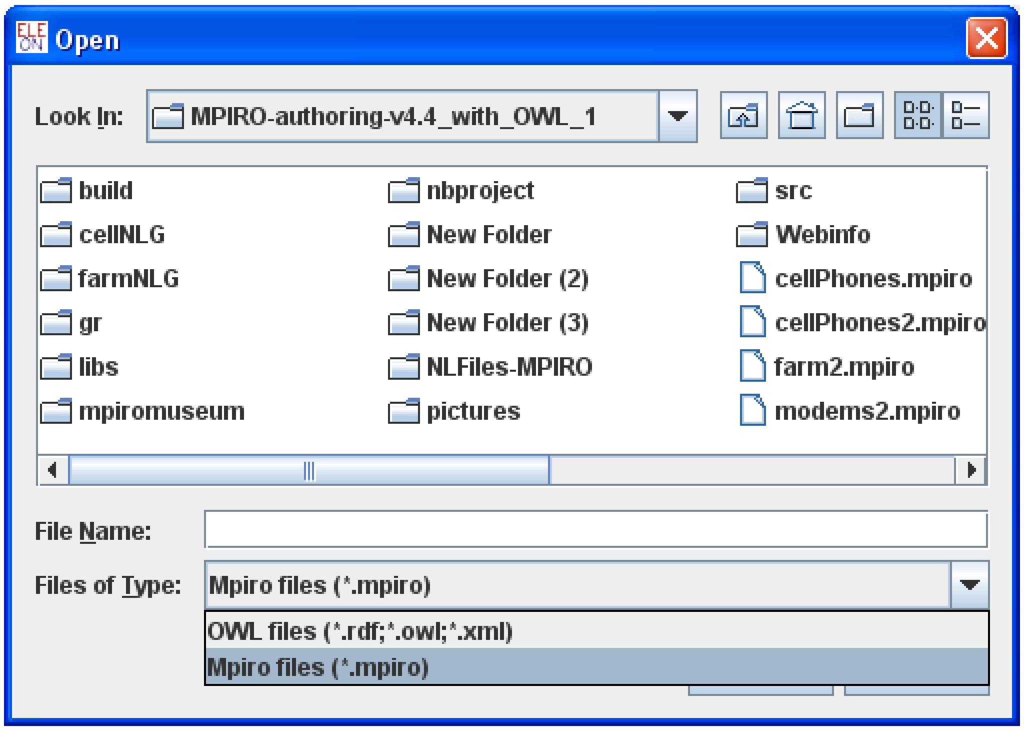
DATA TYPES

DESCRIPTION:
The following data types have been predefined: *string*, *number*, *date* and *dimension.* These are some basic data types that can be used as filler types in addition to the filler types that are inherited from super types.

Save/Load ELEON data

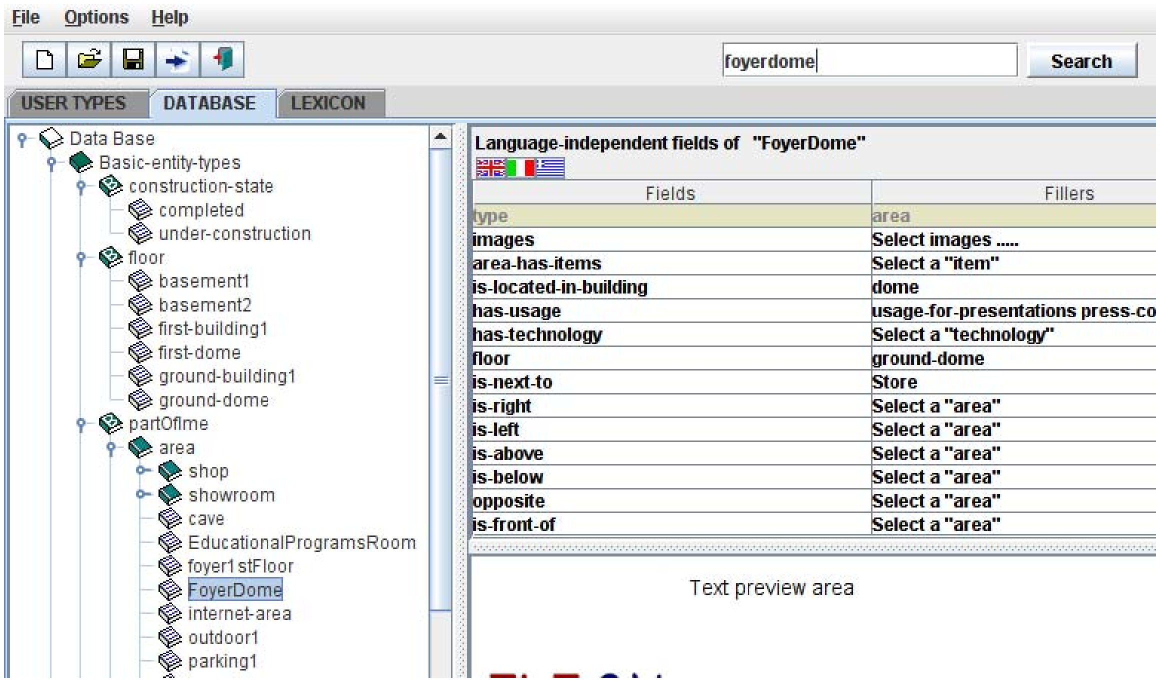
Description:
Eleon can save and load data in its native format.

Examples:



Other functions

Description
Eleon performs some very useful support functions; in particular there is *search* facility, whereby authors can look for an occurrence of a (basic) type, a subtype, or a (generic) entity. Looking for the next occurrence of the same item, involves activating again the search facility. The *help* function provides some pieces of basic information about ELEON, such as current version.



Language-independent fields of "FoyerDome"

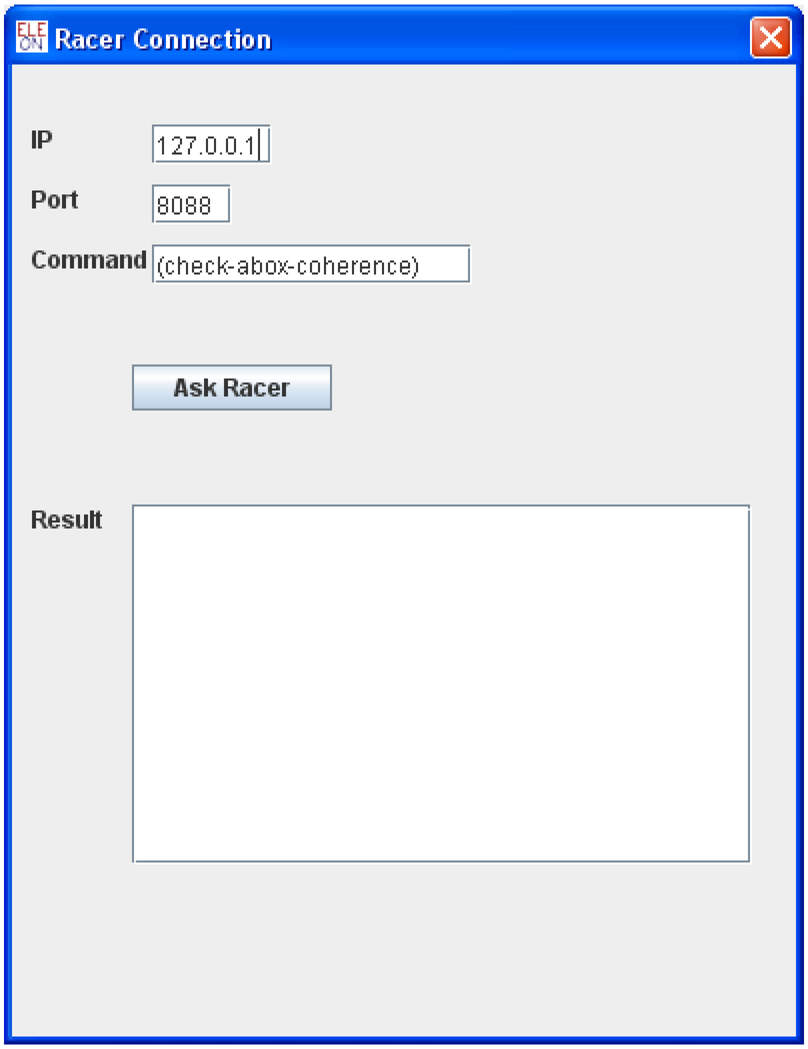
Fields	Fillers
type	area
images	Select images .....
area-has-items	Select a "item"
is-located-in-building	dome
has-usage	usage-for-presentations press-co
has-technology	Select a "technology"
floor	ground-dome
is-next-to	Store
is-right	Select a "area"
is-left	Select a "area"
is-above	Select a "area"
is-below	Select a "area"
opposite	Select a "area"
is-front-of	Select a "area"

Add-ons

NLG Engine
Currently the natural OWL engine is

Consistency Checker (OWL reasoned and Inference Server)
The "Run Reasoner" shows the Racer Connection panel (see next figure). This panel provides an interface to the reasoning engine, RacerPro, via the TCP protocol. In order to get connected with the Racer Server, you have to complete the IP address of the machine in which the server runs and the appropriate port. Then you can insert a command as it is

specified in the Racer user manual. Using this inference service, you can check the consistency of the ontology, retrieve individuals of a particular concept, retrieve the fillers of a property for an individual, etc.



IP 127.0.0.1

Port 8088

Command (check-abox-coherence)

Appendix

Eleon supports the authoring of Ontologies that follow OWL Lite[2] and it also supports the following features from OWL DL[3]: *owl:hasValue, owl:maxCardinality, owl:minCardinality, owl:Cardinality*

[2] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3
[3] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s4

References

1. Dimitris Bilidas, Maria Theologou, Vangelis Karkaletsis: Enriching OWL Ontologies with Linguistic and User-Related Annotations: The ELEON System. ICTAI (2) 2007: 464-467