**D4.1.1**

**Knowledge Base Design document**

ModelWriter

Text & Model-Synchronized Document Engineering Platform

Work Package: WP4

Task: T4.1 – Knowledge Base Design

Edited by:

Erhan Mengusoglu <erhanmengusoglu@mantis.com.tr> (Mantis)

…

Date: 02-Jun-2015

Version: 1.0.0

Apart from the deliverables which are defined as public information in the Project Cooperation Agreement (PCA), unless otherwise specified by the consortium, this document will be treated as strictly confidential.

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author(s) | Date | Remarks |
| 0.1.0 | Ferhat Erata  Moharram Challenger | 30-Apr-2015 | Draft |
| 1.0.0 | Erhan Mengusoglu  Yvan Lussaud | <date> | Initial Release |
|  |  |  |  |

Table of Contents

Contents

[Document History 2](#_Toc430175517)

[1. Introduction 4](#_Toc430175518)

[2. Definition of knowledge base and sample elements 5](#_Toc430175519)

[3. Sample RDF representation of ModelWriter use cases 6](#_Toc430175520)

[4. Representation of ontological structures in the knowledge base 7](#_Toc430175521)

[5. Conclusion 8](#_Toc430175522)

[References 9](#_Toc430175523)

[Appendixes 10](#_Toc430175524)

1. Introduction

This deliverable provides basic design principles for the knowledge base which serves as the repository for metamodels.

* 1. Goal of the Knowledge Base in ModelWriter

The Knowledge base is a master piece of the architecture of the ModelWriter product.

The Knowledge Base has the following functions:

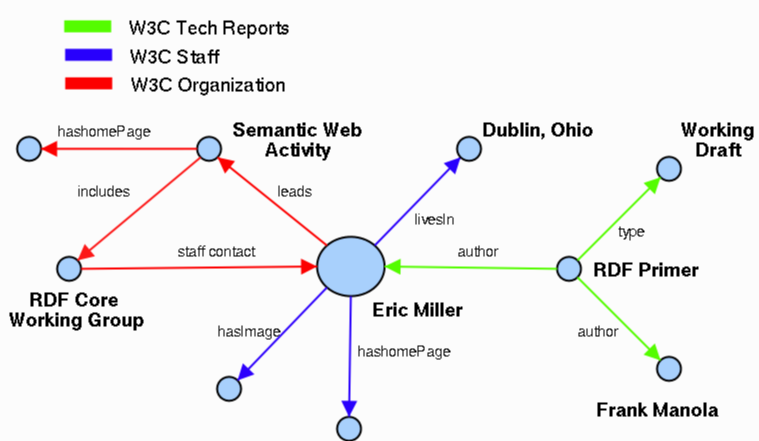
* manage the links between synchronized artefacts(text and models)
* manage …
  1. Acronyms

|  |  |
| --- | --- |
| Abbreviation | Definition |
| RDF | Resource Description Framework |
| WP | Work Package |
| UC | Use Case |

1. Definition of knowledge base and sample elements

Knowledge is defined as “Facts, information, and skills acquired through experience or education; the theoretical or practical understanding of a subject” by the Oxford dictionary. In the context of the project we will take the part “theoretical or practical understanding of a subject” from this definition. In digital environment, knowledge is represented as a network of semantic definition for a particular subject using a semantic web approach.

Semantic web is originally an approach defined by W3 Consortium for creating digitally readable structures for web pages on the internet. This well-defined methodology for representing the data on web pages, later on, found to be useful for representing knowledge in different domains like biology, banking, astronomy etc.

An example semantic web is provided below:

In the project, model elements need to comply the notation of semantic web usually described as Resource Description Framework (RDF) documents. W3C describe RDF structures as “the underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object”.



In this notation, direction of the arc between subject and object is significant.

ModelWriter will use RDF as the meta-model for knowledge-base. By imposing the model elements to comply with RDF notation we will have standardized representation of text documents as models.

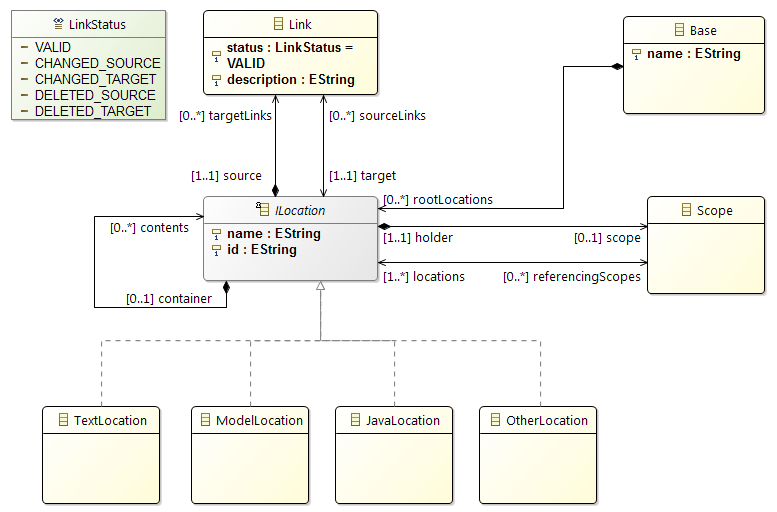
1. Definition of knowledge base to manage Model to Text synchronization
   1. The Knowledge Base Data Structure

The knowledge base data structure is one of those user “hidden-models” (or system models). It will be generated by ModelWriter behind the scene based on links between user inputs: text documents and models.

The Data structure model itself is not meant to be shown, and is derived from:

1. The text obtained from the “Writer” part of Modelwriter.
2. The models obtained from the “Model” part of Modelwriter.

The class diagram of the knowledge base data structure representation is given in the next figure.



In the next section we describe in detail every concept of this representation.

* + 1. The Data Structure Description
       1. The class “Base”

The main class of this representation is “Base”. It contains all root locations (see 3.1.1.2) a root location represents a used resource (e.g. a document, a model, a java class, etc.).

* + - 1. The interface “ILocation”

A location shall have a name and shall also have an identifier.

A location can represent a text location, Model location, java location, etc.

Each location holds its own specific useful information e.g. a text location must hold all references needed to locate the concerned text part.

Each location might be contained by another location. Only locations which are linked together with links and locations which have at least one linked child location are stored in a tree structure.

Every main location might hold a scope (see 3.1.1.3).

Each location can be the source and the target of several links. Target locations must be in a scope (see 3.1.1.3) held by the location itself or by one of its containing locations.

* + - 1. The class “Scope”

The class “scope” represents a set of files (text documents, Models resources, Java code locations, etc.). The goal here is to reduce the creation links only to locations contained directly or indirectly in the scope.

It shall be held by one and only one location and it references a set of locations.

The scope is propagated to all locations contained in the location holder of the scope.

* + - 1. The class “Link”

A link shall have a description explaining the purpose of the link.

A link shall have a source location and a target location.

A link represents a mapping between a source location and a target location. The source location is the holder.

A Link has a status which can be:

1. ***VALID*** – if the link is synchronous and does not present any problem while locating its source and target.
2. ***CHANGED\_SOURCE*** – if the source location has been changed
3. ***CHANGED\_TARGET*** – if the target location has been changed
4. ***DELETED\_SOURCE*** – if the source location has been deleted from the concerned scope
5. ***DELETED\_TARGET*** – if the target location has been deleted from the concerned scope

For a given link, the target location or one of its containing locations shall be referenced by the scope of the source location or one of its containing locations of the same link.

* 1. The Knowledge Base Interactions

This section describes all the interactions in the knowledge base and their related constraints. The possible interactions concern the concepts of the data structure (location, scope and link).

The last sub-section is dedicated to the notification system.

* + 1. Locations related interactions

1. *New location creation*: There is no specific constraint related to the creation of a new location in the knowledge base.
2. *Mark the location as changed*: All the source links and the target links of the concerned location will be marked as changed. In this case the status of the links will be set to ***CHANGED\_SOURCE*** or ***CHANGED\_TARGET***.
3. *Move the location*: This concerns the modification of technical information leading to localize the concerned location. In this case, there is no semantic change; the reconciler component must be able to recalculate the new technical information to localize the concerned location.
4. *Mark the location as deleted*: All the source links and the target links of the concerned location will be marked as deleted. In this case the status of the links will be set to ***DELETED\_SOURCE*** or ***DELETED\_TARGET***.
5. *Delete a location*: The deletion of a location is only possible if it has no source links, no target links and no contained location. In that case, the deleted location must be removed from all the referencing scopes and the scope held by the location to delete must also be deleted.
   * 1. Scopes related interactions
6. *New scope creation*: A scope can be created by referencing one or more locations.
7. *Edit a scope*:
   1. Add a location in the scope: There is no specific constraint related to adding an additional location to the scope in the knowledge base.
   2. Delete a location from the scope:
      1. The deletion of a location from a scope will never remove the location from the model.
      2. The deletion of a location is not possible if the location holding the scope or holding one of its contained locations is the source of a link targeting the location to delete. If the deleted location is the last location in the scope, the scope will be deleted automatically.

The scope deletion is automatically managed by the knowledge base after the suppression of its last referenced location or after the suppression of its holder location.

* + 1. Links related interactions

1. *New link creation*: A link can be created by referencing a source location and a target location. The status of a new link is “***VALID***” by default. There is no other specific constraint related to the creation of a new link in the knowledge base.
2. *Edit a link*:
   1. Reconnect a link to another source location or to another target location. In all cases, the target location of the link must be in the scope of the source location or one of its containing location scopes.
   2. Reverse a link by switching its source and its target. This is a special case of the previous interaction (reconnect a link).
   3. Mark as valid after a location change or deletion. There is no specific constraint related to this interaction.
3. *Link deletion*: There is no specific constraint related to this interaction.
   * 1. Notification system

All interaction leading to a modification of the data structure state shall notify the knowledge base listeners (the reconciler, the synchronizer, the logger, etc.).

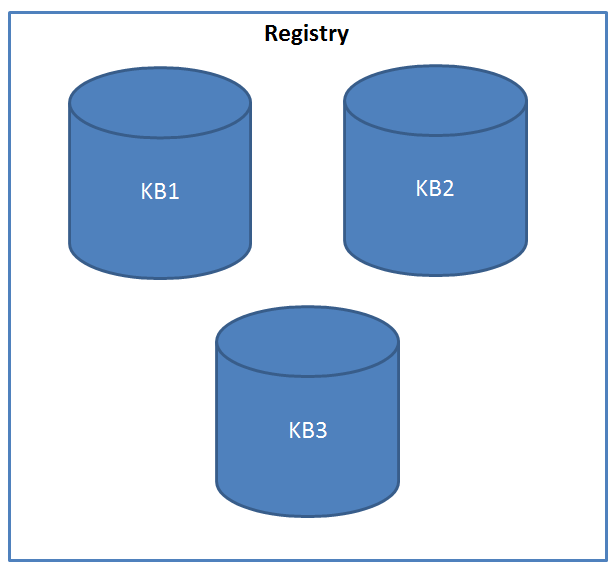
* A listener can be registered to track notifications from a location. The listener will be able to track all the notified change of a location or of one of its contents.
* A listener can be registered to track notifications from a link. The listener will be able to track all the notified change of the link status.
* A listener can be also registered to track notifications from a scope.

The notification system might allow easy filtering notifications (listen only the creation of links notifications, etc.)

* 1. Multiple Knowledge Bases Management
     1. Registry

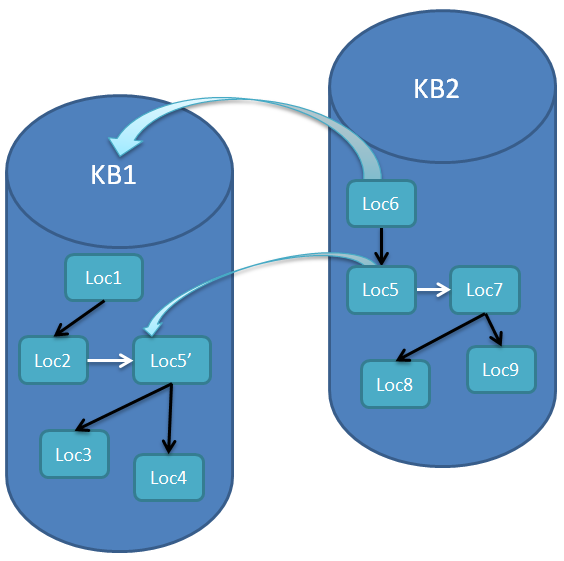
The Knowledge base might need to contact another knowledge base, knowing only the link, but not how to contact it. A registry is needed to provide a service and access the connected knowledge bases.

It will be useful to store links created in the editor.



* + 1. Collaboration

A knowledge base ***KB1*** collaborates in another knowledge base ***KB2*** by offering the access to its locations and links content. The location ***Loc6*** in ***KB2*** represents the ***KB1***. The location ***Loc5*** in ***KB2*** represents the location ***Loc5’*** of ***KB1***. From the ***KB2*** point of view ***KB1*** is like any other resource. It is accessed via a connector dedicated to knowledge bases management. The connector will listen to notification from ***Loc5’*** in ***KB1*** and forwards notifications to the ***KB2***. For instance if ***Loc5’*** is marked as changed in ***KB1***, the connector will mark ***Loc5*** as changed after receiving the change notification in ***KB2***.



1. Use of Knowledge Base in ModelWriter use cases

Knowledge base usage scenarios for each use case to be provided here.

* 1. UC-FR1 – Synchronization between models and documentation (OBEO – Sirius Product)

In this use case we will mainly rely on the model to text synchronization of the knowledge base. The semantic analysis module (LORIA) will be an improvement of the ModelWriter user experience by providing suggestions in the creation the links and also in regard of modification synchronization. The idea here is to ensure the reliability of the link structure and check the semantic analysis on a known scope. This will prepare us for UC-FR2.

* 1. UC-FR2 – Enterprise Architecture (OBEO – SmartEA Product)

In this use case we will use the model to text synchronization of the knowledge base and the semantic analysis. The goal is to rely on the semantic analysis to provide advanced feature in SmartEA. With this use case we will check scalability of the knowledge base.

* 1. UC-FR3 – Synchronization of regulation documentation with a design rule repository (AIRBUS GROUP)

In this use case we will rather focus on the semantic annotation modules; the scenario being to use an OWL file that represents the domain knowledge (Model part) to create links with text. Then the synchronization mechanism based on the KB would be used.

1. Representation of ontological structures in the knowledge base

Ontological structures are represented as RDF documents in the knowledge base.

1. Conclusion

This deliverable will serve as a reference document for designing and implementing model to text and text to model transformations. Bases for synchronizing models with texts and vice versa are also provided in this document.

References

1. Wang, Xiao Hang, et al. "Ontology based context modeling and reasoning using OWL." Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on. Ieee, 2004..
2. <http://www.cs.uu.nl/docs/vakken/b3ii/Intelligente%20Interactie%20literatuur/College%205.%20Context%20Awareness%20en%20Ubiquitous%20Computing%20(Dignum)/Ontology%20for%20contexts%20(verplicht).pdf>

Appendixes

N/A