ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands
T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

D6.2.1- Report Architectural Design-v1.0

ModelWriter

Text & Model-Synchronized Document Engineering Platform

Project number: ITEA 2 13028
Work Package: WP6
Task: T6.2 - Architectural Design

Edited by:

OBEO
Ferhat Erata <ferhat.erata@unitbilisim.com> (UNIT)

Date: 01-March-2015
Version: 1.1

Document History

Version	Author(s)	Date	Remarks
1.0.0	Obeo	01/03/2015	Initial release
1.1.0	Ferhat Erata	15 SEP 2015	Minor Modifications

Table of Contents

Contents

1. Introduction

Role of the deliverable

This document is the first version of the ergonomics guidelines document of the writer part. It may be improved or up-dated depending on the further details and requirements we get from our industrial use case providers.

Structure of the document

This document is organized as follows:

- Chapter 1 introduces the document.
- Chapter 2 describes the ergonomics guidelines
- Chapter 3 WIP-Ergonomics Guideline of the Locations view

Terms, abbreviations and definitions

Abbreviation	Definition
RDF	Resource Description Framework
WP	Work Package
UC	Use Case
MW	ModelWriter

2. Architectural Design

This document (**ADD** for Architectural Design Document) is about the Architectural Design of ModelWriter. It provides the communication scenarios between each ModelWriter component and specifies resulting impacts in term of each component's architecture.
The described architecture defines basic structure and organization for the next research and development Work Packages. It defines also the interactions between:

- "**Model**" Side
  - WP6.1 - Selection and enhancements to modeling tools
  - WP3 - storing and retrieving models from the knowledge base M2M strategies
- "**Writer**" side
  - WP6.2 - selection and enhancements to word processor(s)
  - WP2.1 - storing text meaning in the knowledge base
  - WP2.2 - generating text out of the knowledge base
- "**Knowledge Base**" side
  - WP4 - the repository containing links between text and models including all needed services (ie. semantic/syntactic comparison, consistency checkers, etc.) and all other information defining the knowledge.

As the knowledge base is the starting point of communications between the ModelWriter components, the architecture central composite is dedicated to this knowledge base.

Before illustrating the architectural design details, let us review the objectives of each relevant work package.

3. Review of Work packages Objectives Basing on the FPP Content

Text Parsing and generator - WP2:

This WP objective is to provide the **reversible semantic processor** (parser and generator) **component** of ModelWriter which maps text to formal representations and formal representations to text.
This is broken down into the following sub-goals:

1. Analyze the natural language processing requirements set by the technical documents of the industrial use cases (vocabulary, styles, structures, standards, etc.) so as to identify gaps in existing technology.
2. Define a target semantic representation language for parsing and generation
3. Investigate and compare existing symbolic and statistical approaches to deep semantic parsing and text generation so as to define the best option for ModelWriter
4. Integrate knowledge and constraint rules in a statistical machine learning framework for the semantic processing of higher-level semantic information such as argumentative or discourse structure.

Exploring complementary approaches to semantic parsing, data-to-text and text-to-text generation, the partners involved in WP2 will develop a reversible semantic processor (parser and generator) for ModelWriter based on semantic representations (models) that are rich and precise enough to support both natural language generation and the type of knowledge-based reasoning required by the Industrial Use Cases (e.g., consistency checking and redundancy detection).

Model to/from knowledge Base - WP3:

The objective of WP3 is to provide the **synchronization mechanism** of the ModelWriter platform to keep the "**user-visible models**" consistent with the "**KB-stored models"** and vice versa.
This WP addresses all problems related to the "**model-to-model transformations**" designed by [UNIT + MANTIS] in ModelWriter.

- By "**user-visible models**" is meant those models that have been explicitly created by a Technical Author.
- By "**KB-stored model**" is meant a part of the Knowledge Base devoted to storing pieces of related information, disregarding whether it is represented in user-visible models, in natural-language documents, or in both.

The provided mechanism will be available as an Eclipse-based M2M Transformation Framework which will be extensible enough to accommodate an increasing number of types of (user-visible) models. This will consist of the following main envisioned components which will be developed by [UNIT]:

- **Transformation Manager**: provides the infrastructure to register and launch transformations.
- **Configuration Manager**: for personalizing the behavior of the framework to meet the needs of a specific standard / organization / project / individual.
- **Traceability Manager**: keeps links between elements of user-visible models and elements of the KB.

- **Synchronization Manager**: triggering transformations when synchronization is needed.

Knowledge-Base and Implementation - WP4

This WP objective is to provide the Knowledge Base component (data structures and services) of the ModelWriter platform.
This will be provided by [OBEO + MANTIS + UNIT] and can be broken down into the following:

- Design and serialize the Knowledge Base itself
  The key idea of the envisioned design is that the KB will not consist in just one unique (supposed to be unifying) semantic representation of the contents of a software lifecycle document or model, but will rather consist in a federation of representations cohabiting within the KB:
  KB = federation of parts expressed in various modelling languages.
  For instance, pieces of text representing requirements are better captured as ORM models (WP3's target representation) while other parts (e.g. tables) are better captured by some other formalism.
- Bi-directional text-model synchronization mechanism.
- Services to exploit the KB:
  - Provide Consistency and completeness checks within the same software lifecycle document, allowing automatic quality review of the content (meaning).
  - Provide Consistency and completeness checks between related set of documents and incl. check for compliance to a software engineering standard
  - Provide a Semantic comparison engine between two versions of the same software lifecycle document (i.e. what conceptual changes have happened) in order to reveal what conceptual changes have happened. Coupled with WP2's Text Generation, this feature would allow to automatically generating a Change Log.
- Internal bi-directional synchronization mechanism
  Allowing to detect when changes occur on models of the KB (e.g. through the semantic parsing of a document) or on external models (e.g. through the use of graphical modelers), and providing actions allowing the user to fix these synchronization issues.
  The following features are considered as the minimal sets of features for this synchronization mechanism:
  - detect any change made on a model element of the Eclipse workspace, the Knowledge Base or on a part of the documentation
  - provide specific comparisons engines to work with the KB models (to gain in both comparison quality and efficiency)
  - provide a notification mechanism allowing subscribed components (e.g. the Writer part or a graphical modeler) to be notified of the changes.

  The resulting component will follow the iterations of the ModelWriter software.

- External synchronization mechanism organizing a hierarchy of distributed & collaborating Technical Authors.

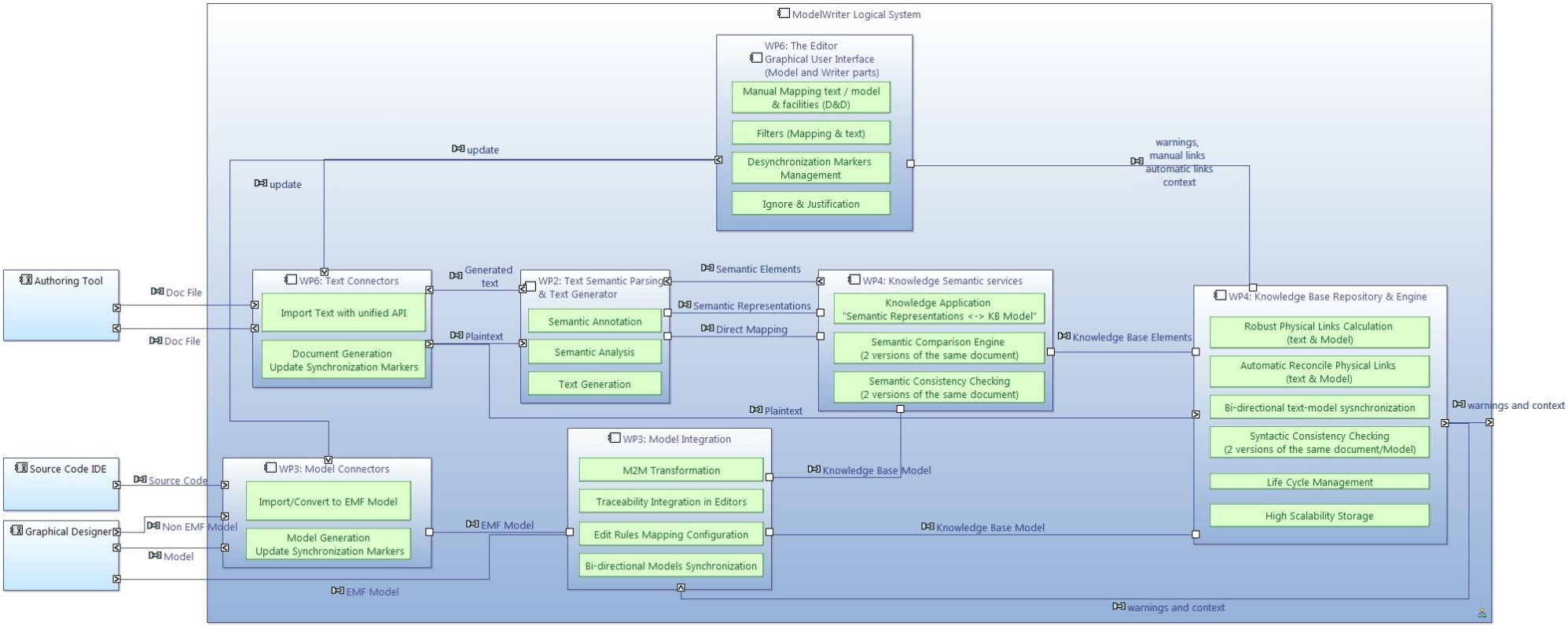
When documentation KB are developed and used within or between organizations, they are built and used collaboratively, by different parties and various stakeholders. The requirements for collaborative aspects of KB design are addressed in this task. The collaboration needs are specified by analyzing the chosen relevant Use Cases involving collaborative knowledge storage and usage taking into account viewpoints of different stakeholders their needs in collaborative working context. The resulting requirements will be used for designing and implementing the synchronization between distant Model Writer's' Knowledge Bases.

ModelWriter Architecture, Integration and Evaluation - WP6

The primary objective of this WP is to assemble and evaluate the successive yearly versions of the integrated ModelWriter. This objective is broken down into the following sub-goals:

- Define the overall architecture of ModelWriter (subject of the current document).
- Provide a unified Graphical User Interface (for both Model and Writer parts).
- Incrementally assemble each major release of the product (starting with a prototype and ending with a final product).
- Evaluate each major release against the WP1's Industrial Use Cases and requirements that were selected as goals for the release.
- Ensure Performance analysis and provide evaluation reports, incl. recommendations for next releases.

4. The Proposed Architecture



ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands
T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

The proposed architectural design is composed of several logical functionalities listed and detailed bellow:

WP6: Text Connectors (Lead by Obeo)

The "**Text connectors**" box represents a bridge from/to the authoring tools to/from the rest of the ModelWriter API:

- The "**Import Text with unified API**" feature (**By Obeo**) will be responsible for importing the text content of a document (from the corresponding authoring tool) and converting it to plaintext using a unified API. This feature communicates with:
  - The **Authoring tools** to get as an input the concerned document file.
  - **WP2 (Text Semantic Parsing & Text Generator)** by providing the plaintext.
  - **WP4 (Knowledge Base Repository & Engine)** by providing the plaintext directly to the knowledge base repository.
- The "**Document Generation and Update Synchronization Markers**" feature (**By KoçSistem)** will be responsible for exporting plaintext to the authoring tool. It will be also responsible for updating all synchronization markers in order to display information to users directly in the authoring tool. This feature communicates with:
  - **WP6 (The Editor)** by getting update information (ie. warnings).
  - **WP2 (Text Semantic Parsing & Text Generator)** by getting the text that needs to be updated
  - The **Authoring tool** by providing it with an updated document file.

WP2: Text Semantic Parsing & Text Generator (Lead by Loria)

The "**Text Semantic Parsing & Text Generator**" box is responsible for two features:

1. Plaintext parsing based on the *semantic elements* contained in the knowledge base. This includes:
   - A "**Semantic Annotation**" feature (**By Loria**) which annotates plaintext with semantic elements thereby linking text and model to help support (at the WP4 level) the synchronization of text and models.
   - A "**Semantic Parser**" feature (**By Loria**) mapping plain text to semantic representations which can be used for querying and/or reasoning.

   These features communicate with:
   - **WP6 (Text Connectors)** by getting the plaintext to parse;
   - **WP4 (Knowledge Semantic Services)** by getting the semantic elements given by the knowledge base;
   - **WP4 (Knowledge Semantic Services)** by providing the semantic representations (ie. RDF formalism or other) and the direct mappings.
2. The generation of plaintext parts based on *semantic elements* of the knowledge base. This will be realized by the "**Text Generation**" feature (**By Loria and Mantis)**
   This feature communicates with:
   - **WP4 (Knowledge Semantic Services)** by getting the semantic representations (ie. RDF formalism or other) and the direct mappings.
   - **WP6 (Text Connectors)** by providing the generated text;

Note that in the current Architectural Design the "**Text Semantic Parsing & Text Generator**" box represents with the "**Knowledge Semantic Services**" box an independent semantic engine which

will help user to semi-automatically detect and generate knowledge base elements from the plaintext (result of the "**Text Connectors**" box).

WP3: Model Connectors (Lead by Unit)

The "**WP3: Model connectors**" box represents a bridge from/to the source code IDE and Graphical Designers to/from the rest of the ModelWriter API:

- The "**Import or Convert to EMF Model**" feature (**By Unit)** will be responsible for importing information from source code IDE or from non EMF Models and converting it to EMF Models. This feature communicates with:
  - Source code IDE and Graphical Designers by getting the input Models which can be EMF Models, Source code, and so on.
  - The **WP3 (Model Integration)** by providing the corresponding EMF Models.
- The "**Model Generation / Update Synchronization Markers**" feature (**By Unit**) will be responsible for exporting and updating the user models. It will be also responsible for updating all synchronization markers in order to display information to users directly in the eclipse editors. This feature communicates with:
  - **WP6 (The Editor)** by getting the update information (ie. warnings).
  - **WP3 (Model Integration)** by getting the up to date EMF Models.
  - Graphical Designer by updating its contents or inserting warnings information.

WP3: Model Integration (Lead by Unit)

The "**WP3: Model Integration**" box represents the M2M bridge between EMF Models and the knowledge models. It is responsible of all translations needed to code and decode model elements to/from the knowledge base according to the user models. Four features are offered by this bridge:

- The "**M2M Transformation**" feature (**By Unit**) which undertakes the design and implementation of model-to-model transformations taking as input:
  a. the language grammars (i.e. meta-models) of the user-visible models
  b. the internal structure of the Knowledge Base

  This feature will communicate with:
  - **WP3: Model Connectors (By Unit)** by getting the user models
  - **WP4 (Knowledge Base Repository & Engine)** by getting the knowledge base models
- The **"Traceability Integration in Editors"** feature (**By Unit**) which will compute the traceability during the Model Writer's life cycle. This feature targets integration inside the Graphical Editor enabling direct access to both documentation and knowledge base elements. This will help users to:
  a. Navigate and reach the ModelWriter editor directly from the Graphical editors.
  b. See the text parts related to each documented element in the corresponding Graphical editor.

  This feature communicates with:
  - **WP4 (Knowledge Base Repository & Engine)** by getting the knowledge base model and by getting physical links to text parts.
  - The **WP3** itself to target Model elements basing on the knowledge physical links.
  - **Graphical Editors** by providing accesses to the Model Writer editor and to the text parts of documented model elements.
- The **"Edit rules mapping configuration"** feature (**By Unit**) which adapts the behavior of transformations to use-case contingencies. A DSL can be defined to specify the catalogue

of transformation rules that can be applied to each element in the models to transform. This DSL allows establishing a changeable behavior of the transformation depending on the type of the element in the input model and also it makes the transformations configuration management independent of the input/output models. This feature will communicate with:

- **WP3: Model Connectors (By Unit)** by getting the user models
- **WP4 (Knowledge Base Repository & Engine)** by getting the knowledge base models

- The **"Bi-directional Models Synchronization"** feature (**By Unit**). This component will be able to keep the transformed models synchronized, whenever changes occur to the input/output models. It will work in two modes.
    a. It will be able to interoperate with the Bi-Directional Synchronization capabilities developed in WP4, receiving requests from that component to trigger model-to-model transformations.
    b. The Synchronization Manager will explore the related models using the traces that are managed by the Traceability Manager and the synchronization can be requested on demand or triggered automatically.

  This feature will communicate with:
    - **WP3: Model Connectors (By Unit)** by getting the user models
    - **WP4 (Knowledge Base Repository & Engine)** by getting the knowledge base models

WP4: Knowledge Semantic Services (Lead by Mantis)

The "**Knowledge Semantic Services**" box represents the interaction between the knowledge base Repository and the semantic parsing/generator. Three features are identified to do so:

1. "**Knowledge Application "Semantic Representations <-> Knowledge Base Model**" feature (**By Mantis):**
    a. Create the knowledge base model from the semantic representation of any text provided as input.
    b. Extract knowledge base elements from semantic representations.
2. "**Semantic comparison engine**" feature concerns the semantic comparison of two different versions of the same document (**By Mantis):**
   Compare two texts using the semantic representations by passing through their knowledge base model and identification of common model elements.
3. "**Semantic Consistency checking**" feature concerns the **semantic** consistency checking of two different versions of the same document (**By Mantis):**
   This feature allows users to check the consistency of semantics created for any document. Inconsistencies will be highlighted when detected.

WP4: Knowledge Base Repository & Engine (Lead by Mantis)

The "**Knowledge Base Repository & Engine**" box is the central part of the ModelWriter API. It manages all kind of information needed to execute the synchronization, comparisons, links calculation etc. It must contain a representation of the last version of a document content, its current version content, the last version of a model content, its current version content too, the already existing links and their kinds (automatic/manual), etc.
Several features are identified in this part of **WP4**:

- The "**Robust Physical Links Calculation**" feature (**By Obeo)** is responsible for providing physical links to text fragments or to model elements. These links must be as robust as

possible in order to preserve text/Model connections after any kind of modifications. This feature communicates with:
    - **WP6** (**The Editor**) by tracking users actions which link a text fragment to a model element.
    - **WP3** (**The Model Integration**) by computing knowledge base elements representing Model elements and by computing the traceability of user Model elements to all linked text fragments needed to handle with the "**Traceability Integration in Editors**" feature.
- The "**Automatic Reconcile Physical Links**" feature (**By Obeo)** will be responsible for reconciling existing links to text & to model's elements. The reconciliation is essential in ModelWriter to keep constructed links available and valid during document/Model lifecycle. This feature is an internal feature of WP4 and does not communicate with any other work package.
- The "**Bi-directional text-model synchronization**" feature (**By Obeo)** will be responsible for synchronizing text and models at a defined instance during the document/Model life cycle. This will provide warnings in case of asynchronous content in both directions (text and model). This feature communicates with:
    - **WP6** (**The Editor**) by providing warnings and information about the synchronization status.
    - **WP3** (**The Model Integration**) by getting the status of synchronization between EMF Models and knowledge base Models.
- The "**Syntactic Consistency Checking**" feature (**By Obeo)** concerns the **syntactic** consistency checking of two different versions of the same document. This feature is an internal feature and does not communicate with any other work package.
- The "**Life Cycle Management**" feature (**By Obeo)** will be responsible for checking consistency of documents, models, manual links and automatic links. It will manage and coordinate actions in order to keep all knowledge consistent. This feature communicates with:
    - **WP4** by getting the Knowledge base elements.
    - **WP3** by getting the knowledge base Model.
    - **WP6** by getting the displayed synchronization information in the Editor; and the user actions (linking text to model elements, ignoring warnings, validating warnings, etc.).
- The "**High Scalability Storage**" feature (**By Obeo)** will be responsible for allocating resources as much as needed by the knowledge repository. This feature is an internal feature and does not communicate with any other work package.

WP6: The Editor "Graphical User Interface" for Model and Writer Parts (Lead by Obeo)

The "Editor, Graphical User Interface" box is the user visible part of the ModelWriter API.

- The "**Manual Mapping text/ Model & facilities**" feature (**By Obeo)** is responsible for providing the capability of manually linking text fragments to models elements. It will also be responsible for providing user facilities (e.g. D&D tools, etc.) in the user interface. This feature communicates with **WP4** (**Knowledge Base Repository & Engine)** by sending it the linking action information required by the knowledge base to calculate robust physical links and reconcile existing links.
- The "**Filters (Mapping & Text)**" feature (**By Obeo**) will be responsible for proposing filtering facilities to help users find text fragments and existing links to models elements.

This feature is an internal feature of the editor and does not communicate with any other work package.

- The "**De-synchronization Markers Management**" feature (**By Obeo)** will be responsible for providing and managing all de-synchronization markers in the Model Writer Editor. This feature communicates with **WP4** knowledge Repository in order to update the synchronization status basing on the user actions in the editor.
- The "**Ignore & Justification**" feature (**By Obeo**) will be responsible for providing the capability to ignore markers and add justification information in the Model Writer Editor. This feature communicates with **WP4** knowledge Repository in order to update the synchronization status basing on the user actions in the editor.