The OntoREM-Mind Mapper Software for Visualising OWL Ontologies

Rami Zayed
University of the West of England/ United Kingdom
ramizayed@gmail.com

Mario Kossmann
Airbus / United Kingdom
mario.kossmann@airbus.com

Mohammed Odeh
University of the West of England/ United Kingdom
Mohammed.Odeh@uwe.ac.uk

Abstract

The OntoREM Mind-Mapper (OMM) tool was developed at the University of the West of England in cooperation with Airbus in order to enhance and automate important aspects of the Requirements Engineering (RE) process as it is implemented in the Ontology-driven RE Methodology (OntoREM) – a novel, knowledge-driven methodology that has been applied to a number of case studies in the aerospace industry over the last years. The development of the OMM tool addressed opportunities for automation of previously manual interfaces between different tool environments that were used within the OntoREM approach, and as such greatly enhanced both the performance and reliability of OntoREM: process times could be significantly reduced, while errors related to manual data transfers could be eliminated. Furthermore, the visualisation of domain knowledge that is stored in domain ontologies (OWL format) was made possible in a user-friendly and customisable way, even of other types of ontologies that are specified in the OWL format, but are not directly related to the OntoREM approach. Using an MVC implementation for the development of the OMM tool led to a high degree of flexibility and increased the compatibility with different types of mind mapping tools available on the market today.

Key words: Mind-mapping, Ontology, OntoREM Methodology, Requirements Engineering, OWL Language, Visualisation.

1.Introduction

This research project was conducted at the University of the West of England in cooperation with Airbus in order to enhance and automate important parts of the Requirements

Engineering (RE) process as it is implemented in the Ontology-driven RE Methodology (OntoREM). OntoREMis a novel knowledge-driven methodology that has been applied to a number of case studies in the aerospace industry during the last years. OntoREM mainly relies on building and exploiting domain ontologies with the assistance of mindmaps when eliciting, updating and developing the ontologies of specific domains including their requirements. This research applied a reverse-engineering process on the OntoREM meta model to automatically generate OntoREM process and domain ontology mindmaps. Using generated mindmaps, requirements engineers can represent the big picture and the complex structures of the existing ontologies (using the ontology web language OWL) that are defined in accordance with the OntoREM meta model, which significantly facilitates communication with stakeholders during the elicitation meetings that are part of the OntoREM process. Hence, this research contributes to bridging that gap between human thinking and machine processing when developing and maintaining domain knowledge.

This paper sheds light ontoa tool that was developedin order to facilitatethe OntoREM processthroughautomating the generation of the OntoREM domain ontology mindmaps, namely the OntoREM-MindMapper (OMM). The OMM software assistsrequirements engineers in automatically transferring the ontologies specified in a given instance of the OntoREM metamodel to corresponding graphical mindmaps. In addition, the software tool supports the other way of the transfer by enabling the import of the knowledge gathered in mindmap templates by the requirements engineers into the relevant parts of the associated OntoREM instance. A case study from the Airbus Photonics project was utilised to evaluate the OMM software.

In the following the paper will give a brief overview of OntoREM and in particular of the OMM tool in the context of OntoREM. Then, the relevance and intended implementation of derived ontology mind

maps, as well as the identified main use cases of OMM are discussed. Based on the latter, the design of the OMM tool and the interactions between the Model, View and Controller(MVC) layers of the tool are described.

2. An Overview of OntoREM

According to Kossmann et al. (2008) OntoREM is a novel Ontology-driven Requirements Engineering Methodology that essentially depends on a knowledge-driven, as opposed to process-driven, approach to requirements engineering [1]. The OntoREM methodology is the main outcome of the OntoREM research project - a joint project between the University of the West of England (UWE) and Airbus.OntoREM defines a set of workflows and associated activities that are triggered and drivenby relevant available domain knowledge and ontologies in order to enhance the traditional RE process and overcome the problems occurring in the process-driven approaches, such as significant corrective reworks, additional costs and further delays [2].Like any other process, the OntoREM process requires some inputs (i.e. the specification of the Problem Space) in order to generate the desired output (i.e. the specification of the Solution Space). The *Problem Space* comprises relevant knowledge about the domain at hand including its needs and concerns, whereas the *Solution Space* contains the outcome of the OntoREM process, which is the associated requirements specification [2]. A high level overview of the OntoREM methodology is depicted in Figure 1, where the parts marked in red indicate missing pathe Problem and Solution Spaces that have to be developed.

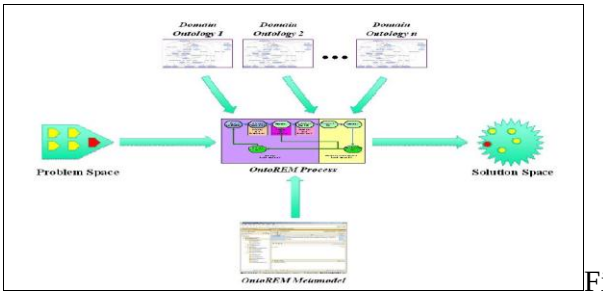

Figure 1: Overview of OntoREM [2]

In order to develop the missing parts of the specifications of the problem and solution spaces, the OntoREM process follows a defined set of workflows and activities, taking into consideration the reuse of any appropriate existing domain ontologies and knowledge bases or, instead, building new ones with the help of relevant stakeholders and domain experts [2]. OntoREM workflows and activities are iterative and can be conducted concurrently, which allows managing any changes or refinements in the domain knowledge/ontologies under development. Unlike traditional RE approaches, the OntoREM process and its defined workflows, with the support of available appropriate domain ontologies, are expected to produce high-quality requirements within shortened time and reduced costs [3]. Figure 2 displays an overview of an example instantiation of the OntoREM process,which is influenced by the traditional steps of the RE process as suggested by Kotonya and Sommerville (1998) (marked in pale green) [4].Figure 2also demonstrates the related supporting software tools that were used in this particular instance of OntoREM and highlights the activities that are expected to be supported by each of these tools.

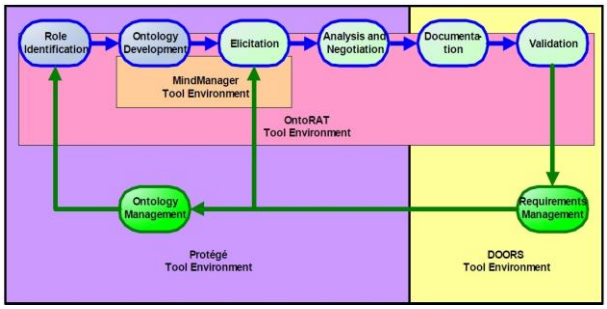

Figure 2: The OntoREM process and its supporting tool environments [5]

3.The OntoREM-MindMapper Tool in the Context of OntoREM

The OntoREM-MindMapper tool enhances the current OntoREM methodology by providing the capability of bi-directional knowledge transfer between the human-readable mind-mapping environment and the machine-readable ontology editing environment while ensuring full integrity of the RE process and associated requirements models [6].

The OntoREM-MindMapper enables the automatic derivation of visual ontology mindmaps from the

OntoREM process and domain ontology meta model. The generated corresponding mindmaps can then be used by requirements engineer to facilitate the maintenance and updates of OntoREM ontologies when new domain knowledge becomes available or changes to the existing domain knowledge are needed, which assists and speeds up the ontology development and elicitation activities of the OntoREM process shown in Figure 2.

In addition, the OntoREM-MindMapper automates the import of the updated domain knowledge that was captured during the elicitation activity and represented in the derived OntoREM mind map templates into the corresponding parts of the domain ontology in owl, which significantly enhances the current OntoREM process by saving the intensive time and effort required to manually perform the data transfers, while reducing the potential for manual errors.

4. Deriving Ontology Mindmaps from the OntoREM Meta model

The activities that were carried out during the RE process of the OMM development were the ones described by Kotonya and Summerville [4]. The RE process started with the Requirements Elicitation activity which was followed by the Requirements Analysis and Negotiation activity. Before documenting and validating the agreed requirements, thorough analyses and studies were conducted to obtain the knowledge required for the reverse-engineering.In parallel with all the other activities, the requirements management activity was maintained concurrently in order to manage and keep track of changes to the OMM requirements during the entire RE process.

A reverse-engineering process was applied to the OntoREM meta model and other ontologies during the RE, specially the elicitation and analysis activities. The main advantageof the application of the reverse-engineeringwasto enable and facilitate the automatic generationof the OntoREM ontology mindmaps from a given OntoREM ontology instance. The reverse engineering process wasessential to obtain a comprehensive understanding of and reliable knowledge about the OntoREMmetamodel, and hence successfully derivecorresponding OntoREM process and domain ontology mindmaps. During the reverse engineering, each part of the OntoREM meta model (i.e. OntoREM process, solution domain ontologies and problem domain ontologies) was analysed and studied individually in order to gain a rich understating of every individual part of the OntoREM meta model, its components, relationships, behaviours and supporting software tools before utilising the gained knowledge to generate corresponding OntoREM ontology mindmaps.This acquiredunderstanding allowed implementingthe OMM tool successfully so that it smoothly interoperates with the existing OntoREM tooling environments while facilitating generating OntoREM mindmaps.

After conducting all the required studies, observations and meetings, the gathered knowledge was utilised to map the OWL elements used in OntoREM ontologies to corresponding mind-mapping elements. This was achieved by defining a set of transformation rules for reverse-engineering OntoREM OWL ontologies to analogous mindmaps.

Using thesemapping rules, the reverse engineering process resulted in a couple of mindmap templates that used different colours, shapes and symbols to map the ontology elements of the OntoREM metamodel. The mindmap templates also specify the structure that the generated mindmaps should follow. Following that, the defined set of transformation rules was implemented in the OMM software to guide the automatic generation of OntoREM ontology mindmaps. Figure 3 shows a high-level view of a mindmap template, its structure and the defined mapping rules.

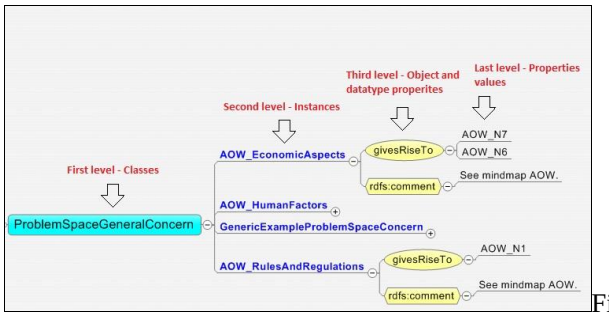

Figure 3: The structure and relationships used to map OntoREM ontology elements to mindmap elements

5.The Use Cases of the OntoREM-MindMapper (OMM) Software

Use-case modelling is an effective technique used during the RE process as a communication means to bridge the gaps between system stakeholders, domain experts and system developers when communicating and eliciting the requirements [7]. With the contribution and feedback of the relevant stakeholders, the OMM use-case model was regularly refined during the RE process until the agreed use-case model was eventually designed. Figure 4 displays theOMM use-case model, consisting of five core use-cases and two main actors, showing all the functionality that the OMM tool shall provide in addition to the system boundaries and its interactions with the actors.

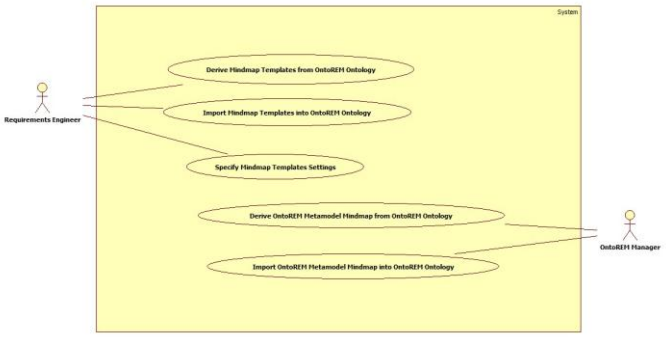

Figure 4: A high-level use case model for OntoREM-MindMapper (OMM)

In the following, three example use cases are described:

- *Derive Mindmap Templates from OntoREM Ontology*:This use case represents one of the two main functionalities of the OMM tool, i.e. exporting ontologies and deriving corresponding mindmaps. The key aim is that the tool should successfully generate visual mindmaps from a given instance of the OntoREM metamodel. These mindmaps should accurately reflect the ontologies already stored in instances of the OntoREM metamodel, in order to provide the Requirements Engineer with a human-readable means to discuss, analyse and capture relevant domain knowledge together with participants in the requirements development process. Three different parts of a given domain ontology should be exported and visualised in mindmap format: (1) the instance of the OntoREM process ontology; (2) the domain's problem space ontology; and (3) the domain's solution space ontology.

- *Import Mindmap Templates into OntoREM Ontology:* This use case represents the second main functionality of the OMM tool, i.e. importing mindmaps and updating related domain ontologies. The aim is that the tool should successfully import the information that was elaborated using the mindmap templates created by the OMM tool into a given instance of the OntoREM metamodel. Doing so, it is vital that data integrity be maintained. The defined mindmap information templates should be used to update the above mentioned three ontologies respectively.

– *Specify Mindmap Templates Settings:*This use case represents the means by which the requirements engineer can update, change, or specify the mindmap templates of the OMM tool. There are two different mindmap templates defined to map ontology elements to mindmap elements (OntoREM Ontologies Mindmap Templates and OntoREM Traceability Mindmaps). This functionality allows the requirement engineer to change the look-and-feel of the mindmap templates and therefore customise the elements' mapping that is used by the OMM tool (i.e. the shapes, colours and font styles). The new changes to the mindmap templates will be applied to both export and import functionalities.

6.The Design of the OntoREM-MindMapper (OMM) Tool

The architectural design is the initial step of the software design process and it builds a critical bridge between the design and requirements engineering processes; it clarifies all the important design aspects early in the design process [8]. Figure 5 shows the architectural design plan of the OMM tool for the entire design process, and established a structural

framework for OMM components and how they interact.

According to Bosch (2000), the system architectural design has a significant effect on whether the system will meet critical requirements such as performance, maintainability, robustness and reliability, which are all considered as non-functional requirements [9]. The OMM architecture was designed mainly based on the non-functional requirements yet it reflected both the functional and non-functional requirements of the OMM tool. To meet such demands, the design of the OMM architecture was driven by both the end-user's priorities of the non-functional requirements and the core use cases specified in the use case model.

Maintainability was of high priority to the relevant stakeholders as the OMM tool was expected to be extended and changed to be in line with the constantly growing mind-mapping business. In order to satisfy the maintainability requirement, the OMM architecturewas designed using fine-grain self-contained components and the MVC pattern was adopted to decrease the dependency between the system components and make the maintenance, change and replacement of any particular component easier, while reducing the influence on the other components.

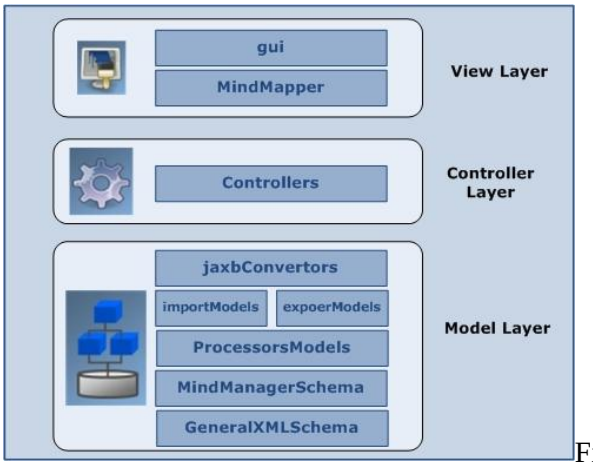

Figure 5: OntoREM-MindMapper (OMM) MVC architecture design

As shown in Figure 5 above, the OMM architecture consists of three separate layers: The business layer (Model Logic), the display layer (View Logic) and the input control layer (Controller Logic). Each layer consists of interactive components and interacts only with its adjacent layers. The layers responsibilities as well as the definitions of the terms Model, Controller and View are explained as follows:

- **Model**: is a layer that defines the business logic. The model encapsulates data structures from the problem domain and provides the methods required to access and modify this data. Models are the representations of real-world objects and processes in software [10]. In the OMM software, this layer is responsible for:

  - Uploading OWL ontology files to the system and creating trees of Java objects (i.e. Java representations) that represent all the elements of the uploaded ontology as Java objects in the memory.
  - Parsing, retrieving, changing, or processing the state of the OWL ontology uploaded using Protégé APIs.
  - Un-marshalling the mindmap files selected by the user and creating trees of Java objects that represent the elements of the selected mindmaps as Java objects in memory.
  - Generating visual mindmaps for the uploaded ontology that represent the ontology metamodel, classes, subclasses, properties, etc.
  - Importing validated mindmaps into the corresponding parts of the ontology uploaded.

- **Controller**: is a piece of code that calls the model methods, controls the data processing and passes the results back to the view. The controller handles the user's interactions such as button clicks and directs the model to manipulate the data accordingly. The controller also modifies the view representations to display the results back to the user [10]. In the OMM software, this layer is responsible for:

  - Rendering the user interactions from the view and determining the needed system behaviours.

- Calling the appropriate methods in the model according to the user inputs and interactions to generate the required outputs.
- Updating the view according to the changes to the model data in order to display the results to the user.

➢ **View**: is what the user interacts with. It specifies how the data is displayed. View representations change according to the changes to the model data [10]. In the OMM software, this layer is responsible for:

- Representing the state of the model data and displaying this data to the user.
- Responding to the user interactions by updating the view (data representation) according to the model changes.

7. The Interaction between MVC Layers in the OntoREM-MindMapper (OMM)

There are different variations ofthe MVC architecture which partially vary in their implementations while holdingthe same three main concepts: model, view and controller. In the OMM software, the architectural MVC design adopted holds no direct communication between the view and the model layers. Instead, the controller is the component that monitors the communications between the view and the model and updates the view whenever the model changes and vice versa. This variation is described as a Passive MVC Model [11].

What differentiates this version of MVC from the traditional MVC is that all communications between the view and the model have to pass through the controller; the controller layer is responsible for communicating the user interactions from the view to the model and the data changes from the model back to the view [10]. Hence, the controller comes in between and manages both directions of the data flow between the view and the model.This design is a recent implementation of the MVC and is common in Apple Cocoa Framework [10].The advantages of adopting this modified MVC are the following:

- It helps in separating the model from the view more completely.
- The controller does not have to be tied to only one pair of view and its associated models; it can direct one or more models while updating one or more views [10].
- It works in the same way as the 3-tier architecture works (i.e. in a linear way): all the communications must pass the controller (the middle ware), which supports distributed systems where the applications are run on separate platforms [11].

Figure 6 shows the adopted implementation of the MVC architecture in the OMM tool, where the controller is the responsible for both changing the data in the model objects and updating the related representations in the view objects. When a user interacts with the system, the communications between the three layers would be as described in the following scenario:

1. The view recognizes the action communicated by the user.
2. The view triggers the appropriate method(s) on the controller.
3. The controller calls the appropriate method(s) in the model to respond to the user's action.
4. The changes to the model data are sent back to the controller.
5. The controller updates the view to display the results/new changes to the user.

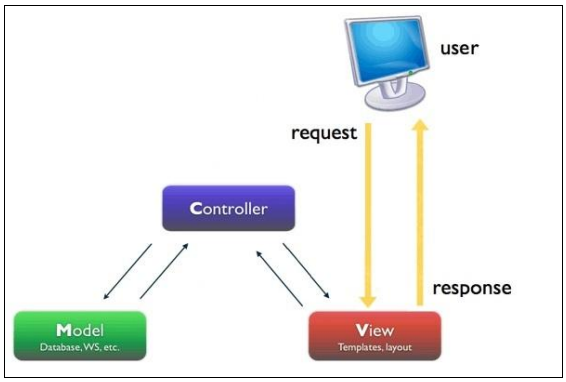

Figure 6: The MVC implementation adopted in the OMM development

8.Conclusion

The paper gave a brief overview of the Ontology-driven Requirements Engineering Methodology (OntoREM) and in particular of the OntoREM Mind-

Mapper (OMM) tool in the context of OntoREM. The relevance and intended implementation of derived ontology mind maps, as well as the identified main use cases of OMM were discussed. Based on these, the design of the OMM tool and the interactions between the Model, View and Controller (MVC) layers of the tool were described.

The development of the OMM tool addressed opportunities for automation of previously manual interfaces between different tool environments that were used within the OntoREM approach, and as such greatly enhanced the performance and reliability of OntoREM. Not only could the process times be significantly reduced because the previously necessary manual data transfer from mind maps into ontology web language (OWL) format was no longer needed; but also, manual errors related to this manual data transfer could be eliminated.

Furthermore, the visualisation of domain knowledge that is stored in domain ontologies (OWL format) was made possible in a user-friendly and customisable way, which brought about the capability to visualise other types of ontologies that are specified in the OWL format, but are not directly related to the OntoREM approach.

Finally, the fact that the OMM tool was developed using an MVC implementation led to a high degree of flexibility in terms of the mind mapping tools that can be used within the OntoREM methodology; and increase the likelihood that the OMM tool will be compatible with future versions or new software of this type of mind mapping tools.

9. References

[1] Kossmann, M., Odeh, M., Gillies, A. and Wong, R. (2008) *From Process-driven to Knowledge-driven Requirements Engineering Using Domain Ontology*. IS'08 INCOSE.

[2] Kossmann, M. and Odeh, M. (2010) *Ontology-driven Requirements Engineering: A case study of OntoREM in the aerospace context*. INCOSE Conference, International Symposium. Chicago: USA.

[3] Kossmann, M., Odeh, M., Watts, S. and Gillies, A. (2009) *Ontology-driven Requirements Engineering with Reference to the Aerospace Industry*. Applications of Digital Information and Web Technologies, Second International Conference. pp. 95-103.

[4] Kotonya, G. and Sommerville, I. (1998) *Requirements Engineering: Processes and Techniques*. Wiley: Chichester.

[5] Kossmann, M. (2010) *OntoREM: An Ontology-driven Requirements Engineering Methodology Applied in the Aerospace Industry*. Ph.D. Research. The University of the West of England: Bristol.

[6] Zayed, R., Kossmann, M., Odeh, M. (2013) *Bridging the Gap between Human Thinking and Machine Processing in Developing and Maintaining Domain Knowledge.*[to be published inINCOSE Conference]

[7]Odeh, M., Hauer, T., McClatchey, R., Solomonides, A. (2003). *Use-Case Driven Approach in Requirements Engineering: the MammoGrid Project*. In: Hamza MH, editor. Proceedings of the 7th IASTED Int. Conference on Software Engineering & Applications, ACTA Press, pp. 562-567.

[8] Bass, L., Clements, P. et. al. (2003) Software Architecture in Practice. 2nd ed. Boston:Addison-Wesley.

[9] Bosch, J. (2000). *Design and Use of Software Architecture*. Harlow: Addison-Wesley.

[10] Eckstein, R. (2007). *Java SE Application Design with MVC* [Online]. Oracle Technology Network Articles. Available from http://www.oracle.com/technetwork/articles/javase/mvc-136693.html [last access 7/6/2015]

[11] Burbeck, S. (1997). *Application Programming in Smalltalk-80: How to Use Model-View-Controller (MVC)* [Online]. University of Illinois in Urbana-Champaign Smalltalk Archive. Available from http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html [last access on 10/6/2015]