ABSTRACT

Open Use Case Modeling Framework (OpenUMF) is a research prototype being developed at the Research Group Software Construction of the RWTH Aachen University offers Use Case based requirements modeling capabilities. Since, it needs to extend new functions for their Eclipse-based application. Domain modeling is a detailed technical analysis after requirement engineering. It can be used in detailed analyses of behavior and the user interface. It is a formalized description of the domain entities and relations. These project aims to scope the domain model into domain specific modeling (DSM) by using Eclipse Modeling Framework (EMF) for create a meta-model and Graphiti as graphics development framework. Expected outcome of the discussion is a domain model graphical editor combine as a part of OpenUMF including evaluation of and problems encountered during the course of implementation.

KEYWORDS—Domain-Specific Modeling, Domain Model, Eclipse Modeling Framework, Graphical Editor, Graphiti.

I. INTRODUCTION

OpenUMF [5] contains Use Case diagramming, textual Use Case details modeling and GUI Sketching functionality. However requirement modeling need domain model to formalized description of the domain entities that are handled by the application and their relationships.

The essential of the project focuses on an implementation of Domain model editor. In the meantime, this paper should evaluate the concepts of domain model via DSM [2] by using of EMF and Graphiti as graphical framework applied for Open UMF platform respectively.

II. DOMAIN MODEL

Domain model is a formalized description of the domain entities that are handled by the application and their relationships. It helps the detailed technical analyses after requirements engineering and can be used in detailed analyses of the behavior and the user interface.

Representation of domain model in Unified Modeling Language (UML) is shown as class diagram [1]. However, this project focusing on of four major parts as follow;

- Entity (main object)
- Attributes (members of entity)
- Operation (features of entity)
- Relationship (association between entities)

![Fig. 1 An example domain model diagram](image)

Figure 1 is an example of domain model showing the relationships between the entities in a simple order domain, including inheritance relationships between customer, corporate customer, and personal customer. Each entity divided as three parts for entity name, attributes and operation in the top, middle and lowers respectively.
Meta-Model in Figure 2, is created using EMF Ecode [3] base on the concept of Domain model above. It is used for creation of Domain Model Editor in Java.

![Fig. 2  ECore meta-model of Domain model editor](image)

III. A GRAPHICAL EDITOR

To visualize the graphical view of meta-model, the Graphical Editor Framework (GEF) and Draw2D [7] provide the foundations for building graphical views for EMF and other model types. Graphical Modeling Framework (GMF) [7] is a tool which encapsulates GEF and Draw2D for creating a graphical editor. However, creating of graphical editor in GMF has been often complex and highly depends on Eclipse plug-in, Java and XML knowledge.

Graphiti [8] is a framework which hides GEF’s complexities from the developer and bridges EMF and GEF to ease and speed up the development of graphical editors.

Basically, communication between user and a Graphiti base tool is happen through screen display, mouse, and keyboard. The process respond the request so-called Diagram Type Agent (DTA) [8] which has to be implemented and made available. DTA provide services and standard for support developer to create graphical editor such as move, resize, delete and so on.

Graphiti can deal with Ecore meta-model also model outside EMF. It is use as data which has to be visualized graphically.

In the rest of this paper, the usage of Graphical Editor, developed with Graphiti framework, and the process of an example of Domain Model Editor creation, are presented.

In Figure 3 showing the classes of Domain Model Editor implemented with the features of Graphiti framework.

![Fig. 3 Domain Model Editor package](image)
In Figure 4 a standard layout of editor is presented. It is split in two sections:

- **canvas**– for presenting and editing diagram
- **palette**– where are the tools for drawing and editing model

The Domain Model Editor, depending on meta-model, visualizes two elements:

- **Connections**– element that visualizes a relation between two entities.
- **Objects**- entity which is a domain object, and can be adding properties and operations also grouping in package.

An editor layout is done by Graphiti DTA, and customizable.

![Fig. 4 A Graphical Editor for Domain Model](image)

IV. CONCLUSION

The implementation of graphical editor is not the new topic but the approach to solving the problem is depending on problem domain. The main step of work is to build a domain specific model which describe problem domain. Next step is to find appropriate graphical framework for building Graphical Editor based on domain specific model. Graphiti is the primary choice with the ease and speed up the development. However, advantages are often followed by a downside. For the editor which is considered as complex, Graphiti proved it’s not because the feature is provide limited.

Future work will be focused on the upgrading Graphical Editor with partially automated code generator a complete solution for creating classes and package designing.

REFERENCE


