Graphical Modeling Environment for Logical User Interfaces Based on Eclipse GMF

Graphical Modeling Environment for Logical User Interfaces
Based on Eclipse GMF

Saehwa Kim
Department of Information and Communications Engineering
Hankuk University of Foreign Studies

This paper presents a graphical modeling environment for pattern and event based logical user interface modeling framework (PELUM), whose implementation is based on Eclipse graphical modeling framework (GMF). Since each of GMF models naturally shows the internal architecture of any graphical modeling environment, we explain our designed GMF models to explain our tool implementation architecture. While GMF is a powerful framework, it has many undocumented or hidden problems that are difficult for novice GMF developers to find out and solve. This paper provides a reference guideline for any developers who want to build their own graphical modeling environment.

1. Introduction

As the requirements for complex UIs for various types of devices are ever increasing, there have been many research activities to provide a model-based user-interface development environment (MB-UIDE) or model-driven engineering (MDE) of user interfaces for multiple devices [1], [2]. Recently, these previous research activities received criticisms about their practicality as in [3], [4], [5]. In [6], we identified some dominant limitations of these approaches and proposed a Pattern and Event based Logical UI Modeling framework (PELUM) to model UIs targeted for multiple embedded systems. PELUM encompasses (1) a pattern-based method for deriving a UI implementation from a UI model, (2) a meta-model for modeling both abstract UI and task model, whose name is Logical User Interface Model (LUM), and (3) its supporting tool.

This paper presents our supporting tool for PELUM, which is a graphical modeling environment for logical UI models. Fig. 1 shows a screen shot of our PELUM tool, which models logical UI model for an Android alarm application. As shown in Fig. 1, PELUM logical

---

1 This work was supported by Hankuk University of Foreign Studies Research Fund of 2010.
UI model is composed of various nodes where some of them, whose name is ‘container’, can contain other nodes. There are also connections between nodes whose name is ‘navigator’. All nodes as well as connections have some graphical symbols and their own labels.

This tool is based on Eclipse graphical modeling framework (GMF). Fig. 2 shows a dashboard that is associated with Eclipse GMF. Eclipse dashboard plays the role of guiding the process of exploiting the associated framework. As shown in the figure, there are six models in GMF: (1) domain model, (2) domain generation model, (3) graphical definition model, (4) tooling definition model, (5) mapping model, and (6) diagram editor generation model. As the arrows of Fig. 2 indicate, the most fundamental model is (1) domain model and the resultant models are (2) domain generation model and (6) diagram editor generation model. Among the remaining models, (3) graphical definition model and (4) tooling definition model are derived from (1) domain model and they can be customized to build our own tool. The remaining model, (5) mapping model is a combination of (1) domain model, (3) graphical definition model, and (4) tooling definition model and it can be also customized to build our own specific tool. The tool binary is build from domain gen model and diagram editor gen model. Usually, these two resultant models do not need to be customized.
As such, our tool implementation is largely composed of modeling four models except domain generation model and diagram editor generation model. Since each of four models naturally shows the internal architecture of our PELUM tool, this paper explains each model one by one to explain our graphical modeling environment. This tool presentation based on Eclipse GMF in this paper provides a reference guideline for any developers who want to build their own graphical modeling environment.

The remainder of the paper is organized as follows. From Section 2 to Section 5, we explain four GMF models, each of which corresponds to the component of the internal implementation architecture for our graphical modeling environment. Section 6 concludes the paper.

2. Domain Model

The domain model is the most fundamental model in GMF. Our domain model exactly corresponds to the PELUM’s meta-model for logical UI model. Fig. 3 shows our Eclipse ecore diagram for logical UI model. While there can be various ways to provide the domain model for GMF, we chose ecore diagram model since it is similar to UML class diagram and thus it is easy to manage. Any diagram model in Eclipse is usually coupled with xml based text model and ecore diagram model is also accompanied with ecore model automatically. These two models are synchronized together automatically: if we modify one model, the other model is...
The most difficult problem in using GMF is that there are various un-documented and hidden error-prone rules as many open-source projects. The most notable problem in designing GMF domain model is that there are preserved names that should not be used for GMF modeling entities in the domain model. Specifically, some words such as ‘model’, ‘component’ should not be used as the names of modeling entities of GMF since they are already used internally in GMF. This is because the name space of GMF internals are not separated with the user models. Since Eclipse problem or warning window or error messages do not say such problems directly, it is hard for novice developers to solve this problem. The reasons why all modeling elements in Fig. 2 have the prefix ‘LUI’, which represent Logical UI, is also to solve this problem. Without this prefix, the naming conflicts occur and thus using GMF is impossible.

3. Graphical Definition Model

The graphical definition model is to define the graphical presentation of each modeling element. GMF provides largely five kinds of graphical presentations: figure descriptor (gallery),
node, connection, compartment, and label, where compartment means a special node that contains other nodes. As shown in Fig. 1, PELUM also provides compartments nodes since some nodes such as appAlarm and listAlarms contain other nodes. Specifically, LUIContainer and its children that are LUIOptions, LUIPopulation, and LUISetting as shown in Fig. 3, are compartments in PELUM. GMF wizard does not guide compartments by default. Therefore, we should manually add these nodes and properly set their attributes as shown in Fig. 4.

As shown in Fig. 1, all nodes have some icon image as prefix of their labels. The current GMF graphical definition model does not provide a way to define icon images for each label. However, there is a way to enable this by manually overriding generated default images by GMF. Specifically, the generated directory in our working space [TopModelName].edit/icons/full/obj16/ contains gif files for each nodes with a file name [NodeName].gif. For example, we replace LUIModel.edit/icons/full/obj16/LUIActivator.gif with our own image file to provide our icon image for node LUIActivator as shown in the left panel in Fig. 4.

Fig. 4. Parts of GMF graphical definition model for PELUM tool.
4. Tooling Definition Model

The tooling definition model is to define the components of the tool palette to model each modeling element. As shown in Fig. 1, the tool palette in the PELUM tool has four tool icons. For this, we have four tool definitions as shown in Fig. 5.

As shown in the tool palette of Fig. 1 and the tooling definition model of Fig. 5, there are only three tools for modeling nodes while there are many nodes as shown in the ecore model of Fig. 3. The reason why we provide such a small set of tools in the tool palette is to simplify the interface for developers. If there are too many tools in the tool palette, it is annoying for developers to use our modeling environment. However, we want all nodes are required to be directly modeled. For this, we need to map each graphical modeling element to a specific tool. This will be defined in the mapping model in the next section.

5. Mapping Model

The mapping model is a combination of the previous three models from Section 2 to Section 4. As shown in Fig. 6, if we click tool 'Non-Instance Presentation', a popup window appears for developers to select one of 'Create LUIInternalService', 'Create
LUINavigator’, and ‘Create LUIPresentation’. In this way, other two tools, ‘Container’ and ‘Instance Presentation’ are mapped with several graphical modeling elements as shown in Fig. 6. All such a mapping is done by in the mapping model.

Besides, the mapping model also defines which modeling elements can be contained in each compartment. Defining this is very tedious process since there is no notion of inheritance in the GMF mapping model. In other words, we cannot reuse a previous definition of the parent node.

The most notable things that require cautions for developers in designing GMF mapping model is that the order of nodes in each model combined for GMF mapping model should be the same in all models. Specifically, the order modeling elements in the domain model, the graphical definition model, and tooling definition model should be exactly the same. Otherwise, the resultant generated tool does not work properly without any warning. This is because the generated editor does not use symbolic identifier but use numeric identifier which corresponds to the ordered number of each modeling entity. This is very error-prone problem of GMF since developers can add or delete any modeling entities in any places of models.
6. Conclusions

This paper presented a supporting tool for pattern and event based logical user interface modeling framework (PELUM) [6], which is a graphical modeling environment for logical UI models. This tool is based on Eclipse graphical modeling framework (GMF). GMF enables users to build a tool binary from user-defined models based on its framework. Fundamental GMF models that users should define to build their own tools are domain model, graphical definition model, tooling definition model, and mapping model. As such, our tool implementation is largely composed of modeling these four models. Since each of four models naturally shows the internal architecture of our PELUM tool, the paper explained each model one by one to explain our graphical modeling environment. As many open-source projects, GMF has various undocumented and hidden error-prone rules, which creates a steep learning-curve and a high barrier for many developers. This paper can be used as a reference guideline for developers who want to build their own graphical modeling environment based on Eclipse GMF.

References

요약: 본 논문에서는 패턴과 이벤트에 기반한 논리 사용자 인터페이스 모델링 프레임워크(PELUM)을 위한 그래픽 모델링 환경을 설명한다. 이 환경은 Eclipse 그래픽 모델링 프레임워크(GMF)를 기반으로 한다. 각 GMF 모델은 임의의 그래픽 모델링 환경의 내부 구조를 자연스럽게 보여주기 때문에, 본 논문에서는 제시하는 도구의 구성 구조를 설명하기 위해 우리가 설계한 GMF 모델들을 설명한다. GMF가 강력한 도구이기는 하나 초보 GMF 개발자가 발견하고 풀기에는 어려운, 문서화되어 있지 않거나 숨어있는 문제들이 많이 있다. 본 논문은 자신의 그래픽 모델링 환경을 개발하고자 하는 개발자들에게 참조할 수 있는 지침을 제공한다.

2 본 논문은 2010년도 한국외국어대학교 학술연구비 지원에 의해서 연구되었음