A Prototype driven approach for the migration of legacy systems to current Eclipse technology

Pholkit Manakitiwiphat   Kamol Limtanyakul

The Sirindhorn International Thai-German Graduate School of Engineering

King Mongkut's University of Technology North Bangkok

1518 Pracharaj 1 Rd., Bangsue, Bangkok, Thailand

Veit Hoffmann   Horst Lichter

Rheinisch-Westfälische Technische Hochschule Aachen Univercity

Software Construction

Ahornstraße 5552074 Aachen, Germany 0241 8021340

Abstract—The KISTERS AG is a German company which provides software solution such as for the energy markets, for environment protection and for management of water and air. Since, it needs to extend new functions for their Java swing-based application. Unfortunately, this legacy technology has insufficient beneficial tools compared with a RCP technology, which offers such as GUI creating framework. Consequently, KISTERS has come up with ideas of both migrating several of their existing Time series management (TSM) Client Applications, utilized for controlling, such as, hydrological and environmental systems, onto the Eclipse platform and reusing beneficial functions by merging them from existing applications.

The project aims to apply a framework for requirement gathering and a prototyping approach in order for collection application requirements and incremental exploration a software development respectively. Expected outcome of the discussion is a new standalone TSM Client Application (TCA) represented as, open plug and play architecture, an extensible application providing a standard to any following existed TSM client applications including evaluation of and problems encountered during the course of implementation.

Keywords—Eclipse RCP, Window builder, UI sketching, Requirement gathering, Evolutionary prototyping, Developing approach, Prototype migration,

I.   Introduction

In fact, almost TSM applications, currently developed at the KISTERS, already operate on the eclipse platform. However, some application implementation previously applied another technology before a widely used of this famous software development environment. In consequence, KISTERS has made a decision to migrate their existing TCA graphic user interface (TCA GUI) onto the eclipse platform.

The essential of the project focuses on an extensibility architecture of TSM application developed with the eclipse RCP (Rich Client Platform). In the meantime, this paper should evaluate the concepts of a prototype driven as a design approach and an openUMF as a RE approach applied for a KISTERS platform.

II.   Related concept (openUMF)

OpenUMF is described as a lightweight approach to analyzing a requirement model. It is commonly applied to a small to medium scope of a project. This requirement collecting framework behaves as an iteration which slightly increases its characteristics. The use case approach, an activities simulation and an explorative prototyping are the essential procedures which allow stakeholders to validate, elect and participate in making decision on the requirements easily.

III.   Methodology

This section discusses implementation aspects of software engineering approach used in this project. Essences are both a modulation of the openUMF for enhancing a software development process and a discussion regarding techniques of software designing, advantages of them included.

222

UACEE International Journal of Advances in Computer Science and its Applications – IJCSIA
Volume 3 : Issue 2   [ISSN 2250 – 3765]

Publication Date : 05 June 2013

A. Adopting of openUMF

As two essential purposes of openUMF are use case driven and user interface prototyping, the stakeholders or customers consequently can easily analyze the requirements. Most procedures have paraphrased from the original document. However, openUMF needs to be adopted for the project setup. As opposed to the standard of openUMF, the user interface prototypes in the TCA GUI project are not only used for requirement gathering, but also used as a basic for evolution prototype.

Figure 1 shows the adopted openUMF approach for the TSM project. This model consists of four major parts as follows;

- Scoping
- UI sketching
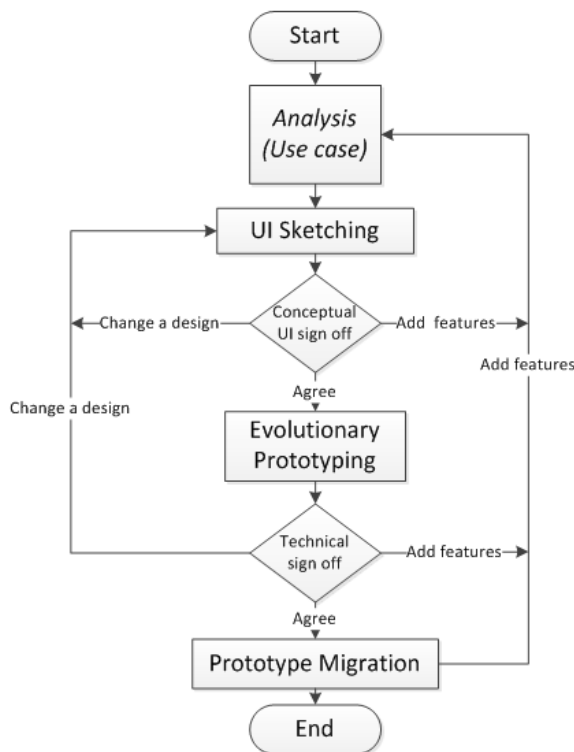- Evolutionary prototype
- Prototype migration

Figure 1. TCA GUI developing approach

The scoping phase mainly defines requirements and plans. The UI sketching phase is a user interface sketching and storyboard. This focuses on techniques of designing in which the stakeholder can evaluate the design easily. Thirdly, evolutionary prototyping is

used to explore a UI quickly with minimal programing effect. The prototype migration phase mainly focuses on the transformation the prototype to the actual application.

B. Scoping phase

In this project, openUMF is used as a framework in order to setup the project and gather the customer requirements. After the functional requirements have been collected, the role of software engineering is to explore and to transform primary information into a vision, business case and an initial use case in order to propose to stakeholders. After that, informal review is performed. The stakeholders validate the project scope until all significant functions are described. If those functions are missing or wrong, so the documents are rejected, they must be reanalyzed again. See Figure 2.
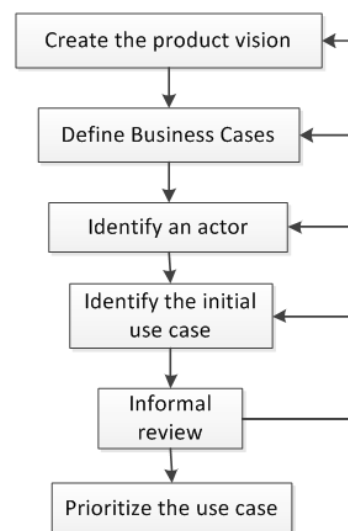

Figure 2. Scoping phase

C. UI sketching

In order to make the same UI overview for each meeting participant, visual UI examples are recommended rather than texture documentation. A graphic user interface helps users quickly reach a common understanding. These interfaces can be designed with the help of a visual tool, sketching on the white board or even on paper.

In TCA GUI project, paper prototype and storyboard have been selected for designing the user interfaces. In other words, a storyboard of almost UIs

UACEE International Journal of Advances in Computer Science and its Applications – IJCSIA
Volume 3 : Issue 2   **[ISSN 2250 – 3765]**

Publication Date : 05 June 2013

in this projecthas been drawn before prototyping. Consequently, all project stakeholders have a clear design proposal. In addition, the UI also can be designed during the discussion.

D. Evolutionary prototyping

At this point, all significant user interfaces and features should be clarified for developers and customers. Developers should know the design of their implementation and functions, whereas customers should be informed of the application appearance (TCA GUI).

However, even though storyboard sketching is provided, some aspects of the solution, such as inline validation and autocomplete, can only be evolved on the real platform. In this case, evolutionary prototyping is suitable for project designing.

By rapidly evolving a prototype application, customers can easily validate these design aspects, and vice versa developers can avoid risks in case that the whole design has to be re-implemented.

The reason of applying the prototype is a complicated domain model of TCA GUI contained specific definitions for data objects and complex relations. The mock-up data technique can also dramatically reduce time spent on prototyping because developers do not need to care for specific detail of real data. For example, API connections can be avoided at this stage. Likewise, a complicated foundation of TSM server connection does not need to be defined yet.

Moreover, by using the data binding feature of Window Builder, the mock-up data properties can be easily synchronized with the UI component values.

E. Prototype migration

Beside the UIs, the prototype of this project also provides some important functions, such as internationalization and data navigation. Consequently, prototype was decided to be migrated to a full-fledged application.

After the connection function for the server was added into TCA GUI, all mock-up data also had to be replaced by the actual information from the server provided by API calls. Thanks to the data binding feature of Window Builder, after the declaration of all real data, the components in each UI can be easily and speedily synchronized. At this step, the order of tasks is concerned and depends on their priority.

III. Evaluation

Regarding the platform requirements and new regulation, Eclipse and RCP are the essential platform and foundation for new TSM client applications. However, since those two technologies were forced to be applied, the major evolution focuses on advantages and disadvantages of others, which have been decided to utilize in this project, and the openUMF approach.

A. Evaluation of TSM approach

The special tool, which is the mainly discussion for this section, is window builder that was used to prototype and used to implement the TCA GUI application. This topic provides some ideas on how worth of them for the latter implementation.

1) Window builder with RCP platform

Window builder enables to create the visual components (both SWT and JFace), such as view and editor, quickly by automatically registering component properties and providing component Java class and methods.

In the prototyping phase, window builder performs well as a tool for creating visual UIs with some basic functions onto the actual environment (eclipse). However, using window builder as a design tool still spends more time designing than using actual UI design software.

Moreover, window builder provides the data binding easily and quickly synchronized data object between UI and TSM database. Window builder also contains useful features, such as source code and UI navigation, properties window of UI components and event registration, for modifying and developing. However, the automatic code generation can lead the messy source code and meaningless name and ID for the components.

B. Evaluation of the developing approach

Continue with the evaluation for mechanisms which used to launch the application, there are three interested approaches which are Use case, UI sketching and prototyping. The major discussion in this section is also their advantages and disadvantages which appear in this project.

1) Use case

In TCA GUI, use case is practically helpful for gathering missing requirements and organizing the functional requirements in order to initialize the project scope and implementation. Since this visual diagram is written from a user perspective, all stakeholders, even with various language speakers, can easily understand and focus on propose of the application functions. In addition, use case can later be used to organize a tasks priority and verify completeness of the requirements.

However, use case requires a plenty of experience for launching an appropriate functions definition and covering all functional requirements.

2) UI sketching and story board

By applying the UI sketching to this project, all stakeholders quickly understand the overall application designs. Drawing the UI is flexible for modifying and can be easily changed even in conference. In other words, participants can review the completeness of designs and suggest alternatives before prototyping.

Each function characteristic, such as search dialog and data tree viewer, is also illustrated as a storyboard. It appropriately provides a clear description of function activities satisfied all stakeholders.

However, the storyboard has to be proposed by arranging a meeting. The stakeholders cannot try to understand all function activities themselves since the storyboard can be misunderstand without having a clear description.

3) Prototyping

In this project, some single function, such as localization for object identity, was prototyped almost ten times because stakeholders have various ideas needed to perform on the actual platform for the decision. In fact, almost functions have to be prototyped at least three times.

In this case, evolutionary prototyping with mocking data can avert the complicated domain model and can be quicker create the several alternatives than a normal implementing. This can also help the stakeholders to decide the choices easily with the visual perspectives.

However, since only one alternative will be choose, the others, especially complex functions or fancy alternatives, have to waste a lot of time.

C. Functional evaluation

In conclusion, 15 functions have been finished within this approach and two functions have been merged to this project. Other four functions have been postponed to other developers while only single function has been developed with open issue.

IV. Conclusion

The concept of eclipse plug-in and RCP provide an extension point without having to modify numerous codes. As expected, almost functions have been developed or merged into TCA GUI.

In term of requirement gathering and developing process, openUMF performs well as a guide line of the application exploration. The stakeholders satisfy the procedures and documents which lead multi validation and clear explanation respectively. The TCA GUI development approach, applied both storyboard sketching and evolutionary prototyping, can avoid the risk of misunderstanding, such as difference languages or roles, consequence in the global overview. However, those duplicate tasks can consume a lot of extra time as the discussion.

In term of technology, window builder performs well as a tool for application developing. The features, such as a drag-drop function and a visual data binding, can dramatically enhance the JAVA application, like TCA GUI, into a potential application to promote the use of window builder among developers.

v.   References

[1] Eclipse official website, plug-in concept, 2005,http://www.eclipse.org/documentation/, (January 17, 2013)

[2] L. Vogel, Rich Client Platform tutorial, 2012   , http://www.vogella.de/articles/EclipseRCP/article.html, (May 11, 2012)

[3] L. Vogel, 2012, JFace Data Binding tutorial, http://www.vogella.com/articles/EclipseDataBinding/article.html, (January 18, 2013)

[4] WindowBuilder official website, 2011, user guide, http://code.google.com/intl/de-DE/javadevtools/wbpro/, (March 28, 2012)

[5] Eclipse wiki website, 2012, Definition and mechanism of data blinding, http://wiki.eclipse.org/Category:Data_Binding

[6] Hoffman, 2011, Instruction for openUMF requirements modeling, unpublished

[7] H.-J. Schlebusch, 2010, KISTER GUI Style guide bases on Eclipse-RCP Style Guide, http://confluence.kisters.de/display/platform/1.3.1+Eclipse+RCP, (June 2010)