SOFTWARE PROCESS FOR INTEGRATED PATTERN ORIENTED ANALYSIS AND DESIGN (POAD) AND COMPONENT ORIENTED PROGRAMMING (COP) ON EMBEDDED REAL-TIME SYSTEMS

SIMBA ANAK BAU

UNIVERSITI TEKNOLOGI MALAYSIA

ABSTRACT

Embedded Real-Time (ERT) systems are becoming increasingly necessary, especially in automotive industries. The complexity to manage the system is growing, where some of ERT applications need high dependability requirements. Component Based Software Engineering (CBSE) appeared to be an attractive approach in the domain of ERT system. CBSE could bring advantages to ERT system such as rapid development time, the ability to reuse existing component and ability to compose sophisticated software. Based on these perspectives, this project aims to enable and support the development of ERT systems based on Pattern-Oriented called Pattern-Oriented analysis and Design (POAD) and Component-based called PErsive COmponent Systems (PECOS), by identifying and defining the process of integrated POAD and PECOS Meta model. The advantages of defining the process are to support development of CASE for ERT and to promote software re-use.

ABSTRAK

Kepentingan sistem masa nyata semakin meningkat terutamanya dalam industri automatif. Selaras dengan peningkatan itu, pengurusan sistem juga bertambah komplek, di mana terdapat sesetengah sistem memerlukan kebolehanharapan keperluan yang tinggi. Kejuruteraan perisian berasaskan komponen merupakan satu pendekatan yang lebih menyerlah dalam domain masa nyata. Kewujudannya telah banyak membawa kebaikan kepada sistem masa nyata seperti pengulangan masa pembangunan, kebolehan penggunaan semula komponen dan kebolehan pengabungan perisian yang komplek. Berdasarksn perspektif tersebut, projek ini bermatlamat untuk membolehkan pembangunan sistem masa nyata berasaskan corak yang dipanggil Pattern-Oriented Análisis and Design (POAD) dan berasaskan komponen yang dipanggil PErvasive COmponent Systems (PECOS), dengan mengenalpasti dan mendefinasikan proses gabungan meta model POAD dan PECOS. Terdapat beberapa kebaikan yang dapat diperolehi dengan mendefinasikan process iaitu dapat menyokong pembangunan peralatan CASE untuk sistem masa nyata dan memperkenalkan penggunaan semula perisian.

TABLE OF CONTENTS

CHAPTER 1

PROJECT OVERVIEW

1.1 Introduction

Component-Based Software Engineering (CBSE) is an approach that has been arises in the software engineering community in the last few years. The idea of CBSE is to allow software engineer to reuse existing component in software development process, in order to improve the quality and reduce the cost of software development. Based on this technical concept, CBSE is concerned with the rapid assembly of systems from components where components and frameworks have certified properties and these certified properties provide the basis for predicting the properties of systems built from components (Bachmann et al., 2000).

Component-Oriented Programming (COP) is part of the CBSE. Murthy (2005), define COP as a collection of interacting components that steps through a program and manipulates data. Each component maintains its own share of data and has its own program piece to manipulate it. COP is used to develop software by assembling components. Szyperski (2002), define software component as a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and subject to third party composition. Moreover, component is a program or collection of programs that can be compiled and make executable, which can be assembled with other component, which can be reused as a unit in various contexts (Wang and Qian, 2005). PErsive COmponent Systems (PECOS), Microsoft’s Component Object Model (COM), .Net component from Microsoft and Common Object request Broker Architecture (CORBA) is an example of component technologies.

Besides that, in order to identify the component in the software development, many computer scientist and engineers referred to any building block of software, such as specification, code or design as a software asset (Yacoub and Ammar, 2004). A pattern is one way to express the component in software development, because it consists of building blocks, which are referred as component.

A pattern is introduced into software engineering as a means of exploiting hard-earned experience in the face of common problems and providing engineer with the language to describe and discuss their problems and solution spaces (Hutchinson and Kotonya, 2005). Yacoub and Ammar (2004), define pattern as a problem that frequently occurs in software design and implementation and then describes the solution to the problem in such way that it can be reused. Pattern can be classified into analysis pattern, architecture pattern and design pattern. Analysis pattern (Yacoub and Ammar, 2004) is analysis that involves looking behind the surface of requirement to understand the problem, architecture pattern (Hutchinson and Kotonya, 2005) is a generative reuse mechanism, featuring in the move from abstract requirement to abstract architectural solution and a design pattern is a design solution to a frequently recurring design problem in a given application domain (Yacoub et al., 2000).

Software pattern is becoming more popular in software development as many of approaches based on pattern were introduced. One of the approaches is called Pattern-Oriented Analysis & Design (POAD). POAD is an approach based on design pattern. POAD is a methodology to design software application using software patterns and to produce pattern-oriented analysis and design. POAD methodology has the capability to glue pattern at high level, also providing logical views to represent application analysis and design as a composition of the pattern. POAD provides a structural approach to use design patterns as building blocks in designing application (Yacoub and Ammar, 2004).

A pattern-oriented is an attractive goal applies into CBSE, because of the appearing similarities in what they try to achieve, such as time complexity, cost effective, and high-quality software. Moreover, the combination of these two approaches (pattern-oriented and component-based) is to solve a similar problem from completely different angles and in such ways that they are likely to be completely incompatible. The benefits of applying patterns to CBSE are (Hutchinson and Kotonya, 2005), 1) the reuse of experience, 2) the development of standard types of solution, 3) a normalized mechanism for abstracting from multiple example in order to extract “best practice” and 4) a means of communication and a method for categorizing problems.

A software pattern has been used in different domains, for example web applications, windows environment and embedded systems. An embedded system is a computer system which is part of larger systems and performs some of the requirements of these systems. Most of these embedded systems are characterized as real-time systems, which consist of real-time properties such as response time and worse case execution time, called Embedded Real-Time systems (ERT) (Crnkovic, 2005). The automobile control systems, industrial processes control systems, mobile phones, or small sensor controllers, are some example of ERT systems.

ERT system is a system whose correctness depends on timeliness and logical correctness, this means that system should satisfy explicit response time constraints or it is assumed as a fail (Crnkovic, 2005). The ERT systems usually have both hardware and software interacting with each other to accomplish a specific task. Hardware tries to satisfy timing constraints, while software reduces the overall cost and provides design flexibility (Jawawi, 2003). One of the characteristic of ERT systems is time constraint, that means a components of the system must be run concurrently and communicate with each other under predefined timing constraints.

1.2 Problem Background

Since the development of ERT systems are becoming increasingly necessary, especially in automotive industries, the complexity to manage the system is growing, where some of ERT applications need high dependability requirement. Therefore, in development of ERT, the system design should fulfill the demanding requirement with respect to limited resources, real-time requirement, reliability cost and also reusability (Crnkovic, 2005). For example, since software was first included in cars about 15 years ago, the amount of embedded code has grown exponentially from around 100 kilobytes to a projected 1 gigabyte in the latest generation of high-end automobiles. As a result, the methods and technologies that have traditionally been used to develop embedded systems are starting to reach the limits of their scalability (Colin et al., 2005)

Moreover, the development of ERT system has to consider non-functional properties because the correct operation of a system is not only depending on the correct functional working of its components but also dependent on its non-functional properties. ERT systems have both non-functional and strict functional requirements. The end to end quality of service (QoS) properties should be ensured in ERT systems such as timeless and fault tolerance (Jawawi, 2003).

CBSE has been used in many application in software engineering such as desktop environment, e-business application, internet and wed-based application (Crnkovic, 2005). ERT is one domain that uses CBSE, in ERT systems CBSE appears to be an attractive approach. CBSE could bring advantages to ERT system (Crnkovic, 2005) such as rapid development time, the ability to reuse existing component and ability to compose sophisticated software. When CBSE applied to ERT systems, it could improve software maintainability, increase software reliability, rapid software development and rational task separation and faster adoption.

The recent trend in software engineering is to combine CBSE with other methods to make CBSE as imperative for ERT development (Colin et al., 2005). Based on this perspective, Universiti Teknologi Malaysia, Skudai has come out with a paper which concerns on pattern oriented and component oriented in order to improve ERT system. In this paper Jawawi (2005) introduced the combination of Meta model Pattern-Oriented Analysis & Design (POAD) together with component model called Pervasive Component Systems (PECOS). In order to improve the quality of software, Yau and Dong (2000) appeared with a paper that concerns on integration a component based in software development with design pattern. Hutchinson and Kotonya (2005) appeared with other paper, which discussed about applying pattern into CBSE.

Based on the review in these three papers, the similarity that can be found is what they try to achieve, which are to promote reused in component-based, in order to improve the quality of software such as complexity, timing constraint and cost.

1.3 Problem Statement

Generally, the integration defined is the combination of methods or approaches, with the expectation of achieving a better performance (Colin et al., 2005). This project is focuses on identifying and defining the process of integrated POAD and PECOS Meta model into formal form. The defined process into formal form is important to enable and support the development of ERT systems based on the two approaches (POAD and PECOS). The advantages of defined process from existing Meta model is to enable to support the development of CASE tools for ERT software and also promote software reused in ERT systems without sacrificing the non-functional requirement such as timeless, predictability and constrained resources.

1.4 Project Aim

The aim of this project is to identify and define software process for integrated of POAD and PECOS Meta model into formal form, with expectation to enable ERT development based on the two approaches, (POAD and PECOS).

1.5 Objectives

The objectives of the project are:

i. To study and identify the software process for integrated POAD and PECOS Meta model
ii. To define the software process for integrated POAD and PECOS Meta model using SPEM.
iii. To demonstrate the applicability of the proposed process using the UML-RT.

1.6 Scopes

The scope of this study will be limited to the following:

i. The analysis and early design of Pattern-Oriented Analysis and Design (POAD).
ii. PErsive COmponent Systems (PECOS) as a component technology.
iii. The implementation only for medium size of Embedded Real-Time systems.

REFERENCES

1. Atkinson, C., Paech, B., Reinhold, J. and Sander, T. (2001). Developing and Applying Component-Based Model-Driven Architectures in KobrA. Proceeding IEEE.
2. Bran Selic, B. and Rumbaugh, J. (1998). Using UML for Modeling Complex Real-Time systems.
3. Crnkovic, I., Larsson, S. and Chaudron, M. (2005). Component-based Development Process and Component Lifecycle. 27th Int. Conf. Information Technology Interfaces ITI 2005, June 20-23, 2005, Cavtat, Croatia.
4. Fowler, M. (1996). Analysis Patterns: Reusable Object Models, Addison-Wesley.
5. Frohlich, P. H. (2003). Component-Oriented Programming Language: Why, What and How. Ph.D. Thesis. University of California.
6. Genssler, T., Christoph, A., Schulz, B., Chris, M., Winter, M., Muller, P., Zeidler, C Stelter, A., Nierstrasz, O. and Schonhage, B. (2002). PECOS in a Nutshell: PECOS handbook.
7. Heineinan G. and Council1 W. (2001). Component based Software Engineering, Addison Wesley.
8. Hutchinson, J. and Kotonya, G. (2005). “Pattern and Component-Oriented System Development”. Proceeding Of the 2005 31st EUROMICRO Conference on Software Engineering and Advanced Applications.
9. Jawawi, D. N. A. (2003). Embedded Real-Time Software. Technical Report 2003. Universiti Teknologi Malaysia, Skudai.

10. Jawawi, D. N. A., Mohamad, R., Deris, S. and Mamat, R. (2005) “Transforming Pattern-Oriented Models into Component-Based Models for Embedded Real-Time Software Development”. Malaysia Software Engineering Conference (MySEC’05).
11. Jawawi, D. N. A. (2006). A framework for Component-Based Reuse for Autonomous Mobile Robot Software. Ph.D. Thesis. Universiti Teknologi Malaysia, Skudai.
12. Kacsuk, P., Dbzsa, G. and Fadgyas, T. (1997). A Graphical Programming Environment for Message Passing Programs. Proceeding IEEE.
13. Kehtarnavaz, N. and Gope, C. (2006). Dsp System Design Using Labview and Simulink: A Comparative Evaluation. ICASSP 2006.
14. Kotonya, G., Sommerville, I. and Hall, S. (2003). Towards A Classification Model for Component-Based Software Engineering Research. Proceedings of the 29th EUROMICRO Conference “New Waves in System Architecture” (EUROMICRO’03).
15. Lepasaar, M. and Makinen T. (2002). Integrating Software Process Assessment Model using a Process Meta Model. Proceeding IEEE.
16. Lu, S. and Halang, W. A. (2005). A Component-based UML Profile to Model Embedded Real-Time Systems: Designed by the MDA Approach. Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA’05).
17. Luders, F., Ahmad, S., Khizer, F. and Dhillon, G. S. (2007). Using Software Component Models and Services in Embedded Real-Time Systems. Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07).
18. Mohamad, R. (2007). Pattern-Oriented Approach for Developing Multi-Agent Based Systems. Ph.D. Thesis. Universiti Teknologi Malaysia, Skudai.
19. Moller, A., Akerholm, M., Fredriksson, J. and Nolin, M. (2004). Evaluation of Component Technologies with Respect to Industrial Requirements. Proceedings of the 30th EUROMICRO Conference (EUROMICRO’04).

20. Murthy, V. K. (2005). High Performance Cluster Computing using Component-oriented Distributed Systems. Proceedings of the First International Conference on e-Science and Grid Computing (e-Science’05)
21. Mark A. Yoder, Bruce A. (2006). Teaching DSP First with LabVIEW. Proceeding by IEEE.
22. Nierstrasz, O., Arevalo, G., Ducasse, S., Wuyts, R., Black, A., Muller, P., Zeidler, C. and Genssler, T. (2002). A Component Model for Field Devices. Software Composition Group, Institute Informatik and Angewandte Mathematik, University of Bern, Switzerland
23. Nierstrasz, O., Genssler, T. and Schonhage, B. (2002). Components for Embedded Software: The PECOS Approach. CASES 2002, October 8–11, 2002, Grenoble, France.
24. Peter Muller, P., Zeidler, C., Stich, C. and Stelter, A. (2001). PECOS - Pervasive Component Systems.
25. Jamal, R and Wenzel, L. (1995). The Applicability of the Visual Programming Language LabVIEW to Large Real-World Applications. Proceeding by IEEE.
26. Riehle, D. and Zullghoven, H.(1996). Understanding and using pattern in software development theory and practice of object systems.
27. Rumpler, B and Elmenreich, B. Considerations on the Complexity of Embedded Real-Time System Design Tasks. Supported by the FIT-IT program of the Austrian Federal Ministry of Transport, Innovation, and Technology, and by the European IST project DECOS.
28. Rumpler, B. (2006). Complexity Management for Composable Real-Time Systems. Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing.
29. Sha, L. (2004). Open Challenges in Real Time Embedded Systems.
30. Software Process Engineering Metamodel Specification (2005). Object Management Group.
31. Wang, A. J. A. And Qian, K. (2005). Component-Oriented Programming. Southern Polytechnic State University Marietta, Georgia. Wiley Interscience.

32. Wang, L. (2005). Component-Based Performance-Sensitive Real-Time Embedded Software. Texas A&M University, College Station, TX.

33. Wan, J. A. (2000). Towards Component-Based Software Engineering.

34. Yau, S. S. and Dong, N., (2000). “Integration in Component-based Software development using design pattern”. Proceeding IEEE.

35. Yacoub, S. M. and Ammar, H. H. (2004). Pattern-Oriented Analysis and Design: Composing Pattern to Design Software Systems, Addison Wesley.