MARMOT AND PECOS HYBRID APPROACH FOR EMBEDDED REAL TIME SOFTWARE DEVELOPMENT

Suzila Sabil¹, Dayang Norhayati Abang Jawawi²
¹,² Department of Software Engineering, Faculty of Computer Science and Information Technology, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
Email: suzy_sbl@yahoo.com¹, dayang@utm.my²

ABSTRACT

Embedded real time (ERT) software development involves multi-disciplinary knowledge such as software, mechanical and electrical engineering. Besides that, multi-constraints such as timing, resources constraint, statically predictable, safety-critical and memory are also important in ERT software development. From this perspective, component based development (CBD) appears to be one of the appropriate approaches to design the ERT software due to the ability of domain experts to interactively compose and adapt sophisticated ERT software. Besides that, it can decrease the development time and improve the software quality. Therefore, existing component model such as MARMOT is introduced to master in multi-disciplinary knowledge and PECOS is good support for multi-constraints extra-functionality requirement. However, MARMOT and PECOS still have their own weaknesses. MARMOT cannot support software component at detail design while PECOS does not provide component modeling. Due to the strength and the weaknesses of the MARMOT method and PECOS component model, this paper propose to hybrid the two approaches to produce a better approach for CBD of ERT software.

Keywords: Component Oriented Programming, Embedded Real Time System, Component Oriented Analysis and Design.

1 INTRODUCTION

The ERT software development typically has multiple dimensions of resource constraints. The essential dimension is time, since many embedded system have real time requirement [1]. ERT system is a system in which the correctness of the system depends not only on the logical results, but also on the time at which the results are produced [2].

Apart from that, the ERT systems usually have both hardware and software interacting with each other to accomplish a specific task. Thus, consideration of hardware and software in ERT system development is needed. An ERT system development involves multi-disciplinary knowledge that includes fields of software, mechanical and electronic engineering. Engineers struggle hard to master the pitfalls of modern, complex embedded systems, and they often only approach the problems from their individual perspectives [3]. Engineers require a software engineering method to assist them in communicating and understanding each other.

CBD is a promising approach for embedded systems typically for multi-constraints such as an essential constraint as timing [1]. Recently, CBD method can support multi-disciplinary knowledge that extended with existing methodology such as MARMOT method extended from KobrA method [3]. The benefits of CBSE for ERT development as described in [4] and [5] consist of the ability to secure investment through reusability of existing component, the ability for domain experts to interactively compose and adapt sophisticated embedded system software, the ability to decrease of development time and improvement of quality.

Existing industrial component model technologies such as OMG’s CORBA Component Model (CCM), Microsoft’s (D)COM/COM++, .NET, SUN Microsystems’ JavaBeans and enterprise JavaBeans, are not suitable to develop ERT systems because they do not address the multi-constraint and multi-knowledge in ERT systems. To meet the requirements of ERT systems, a number of component technologies such as MARMOT [6] and PECOS [7] have emerged. All of the component models have their strengths and weaknesses for their own domain. Based on our previous evaluation [8], we found that PECOS is suitable to support multi-constraint extra-functionality requirements whereby MARMOT method is good to support multi-disciplinary knowledge. MARMOT method is good in component modelling and graphical
component but it cannot support software component at detailed design level. PECOS component technology cannot support component modelling but it good support software component at detailed design and clearly present interfaces for component communication.

Therefore, the main objective of this paper is to hybrid the two approaches between PECOS component model and MARMOT method through the metamodel. These hybrid approaches will enable systematic and detailed modeling approach for CBD of ERT software development.

Section 2 discusses about PECOS component model. Section 3 elaborates about MARMOT method. Section 4 presents the hybrid result of PECOS component model and MARMOT method. The applicability of the result is illustrated with an intelligent wheelchair case study at Section 5. Section 6 contains the conclusion of the paper.

2 MARMOT METHOD

MARMOT (Method for Component-Based Real-Time Object-Oriented Development and Testing) is an extension to the KobrA [9] method. MARMOT adds concepts addressing the specific requirements of developing ERT software. The key activity in MARMOT is composition where it is represented by a hierarchy of components that includes parents/child relationship of the system. The key ideas of MARMOT are information hiding and divide-and-conquer. MARMOT is divided into three types of component which are: 1. Software 2. Hardware which is divided into electronics, mechanics and mechatronics components 3. Software/hardware components.

Software and hardware components are treated in the same logical way. In principle, hardware components in an embedded system typically consist of the hardware itself and a device driver or “hardware wrapper” to communicate with the software. Respectively, software engineer will be interested in software component and mechanical and electrical engineer will be interested with hardware component. To model ERT software using MARMOT, the design is not only made to suit the software requirement but also the hardware requirements. Therefore, MARMOT, which consists of complex metamodel that inherits from KobrA metamodel to fulfil the software and hardware requirement.

MARMOT components follow the principles of encapsulation, modularity and unique identity that most component definitions put forward. Encapsulation will separately divide description of the software into two parts which require description specification to define what a software unit does from the description and realisation is to define how software does it.

Specification is a description of component artifacts that consists collection of every external knowable about a component. The description components artifacts is fully specify a component in a way that can used by the system and assembled in a system. The realization is a description of component artifacts that consists collection of how a component is externally realized. Where, each component within a system can be described through suitable models such as UML diagrams, as if it was an independent system in its own right.

The MARMOT process model includes 4 activities, and they are 1. decomposition 2. embodiment 3. composition 4. validation. Decomposition process will implemented divide-and-conquer concept where the system will be subdivide the entire ERT software into smaller parts that are easier to understand and control. On the other hand, when composition, an individual component has been implemented, or some others will be reused, the system is put together. Embodiment is a concern with the implementation of a system and a move towards more and more executable representations. It turns the abstract system represented by models into more concrete representations that can be executed by a computer. Validation activity is performed to check whether the concrete representations are in line with the abstract ones.

3 PECOS COMPONENT MODEL

PECOS (PErvasive COmponent System) is a component based model for embedded systems. It consists mainly of components communication through ports connected with connections. It provides an environment that supports the specification, composition, configuration checking, and deployment of embedded systems built from software components [10]. PECOS is divided into two main parts, which are static structure model and execution model. Static structure model has three types of component model; active, passive and event component model. Property bundle (scheduling, memory and initialisation) that describes the multi-constrain extra-functionality requirement aspects such as timing or memory usage is associated with this static structure of component. Ports denote data is available to be shared with other components. Connector is used to model actual data sharing between specific ports on specific component.
Execution model defines the behaviour of the component execution. Two different behaviours associated with active and event component: execution behaviour to determine the action that is performed when the component is executed, synchronisation behaviour that is responsible for synchronising the data space of active or event component with that of the parent.

In PECOS, a component is a unit of design that has a specification and a realization. To specify its component, PECOS component technology uses CoCo language for modelling. Developers need to learn the language before they can use PECOS in their CBD process. PECOS supports multi-constraint extra-functionality requirement such as timing specification and timing analysis. Component timing properties in PECOS are specified using properties bundles and Rate Monotonic Analysis (RMA) to analyse the properties.

4 INTEGRATION BETWEEN MARMOT AND PECOS

The mapping result of two metamodels is presented in Figure 1. Discussion on MARMOT metamodel, static Komponent Implementation structure is used as an integration point. It is because component design is performed at embodiment activity in MARMOT. Two strategies involved in embodiment process model in MARMOT are called reuse and implementation. Reuse process model means that it will reuse existing component model while implementation strategy is used to design component from the scratch. However, three classes were removed from original MARMOT metamodel which are quality policy and quality documentation because this two class diagram will be supported using PECOS property bundle component infrastructure. On the other hand, decision resolution model were also removed because in this research we did not consider product line elements. The other components and class diagram in both MARMOT method and PECOS component infrastructure will be maintained in the same relationship with Komponent Implementation class diagram.

Metamodel PECOS will be integrated with Komponent Implementation. It is because KobrA represented by meta-metamodel where PECOS represented by only metamodel. Due to this condition our mapping considerations are: 1. to integrate PECOS metamodel with the simplest element in KobrA such as Komponent Implementation and 2. PECOS metamodel were integrated into a more complex KobrA meta-metamodel.

Overall elements in PECOS will be integrated with Komponent Implementation using same relationship. Based on our work definition, MARMOT is used in early modeling to fulfill multi-disciplinary knowledge requirements of ERT system supported by embodiment process model.

Figure 1. Mapping between MARMOT method and PECOS component model.
Whereby, PECOS is to fulfill multi-constraint extra-functionality requirement of ERT system supported by property bundle at design level. Therefore, an intelligent wheelchair case study is used to model and design the requirement base on the hybrid approaches as proposed.

A prototype of intelligent wheelchair was developed to support our researches in ERT software engineering. It includes two types of requirement which are hardware and software requirement specification. The hardware requirement is represented as shown in Figure 2. Several hardware are used in intelligent wheelchair such as ATMEL MEGA32 is used as a processor, includes four types of sensor and motor. The entire behavior of the software requirement is represented in use case diagram as shown in Figure 3. It includes two actors which are user and obstacle. User or ERT developer will handle three states which receive movement control commands, configure speed and display status. Display status is extended by control behavior and it also includes display battery status. Control behavior also includes control movement. Meanwhile, the actor obstacle will detect obstacles in the surrounding. The interaction between user and the intelligent wheelchair system is represented by interaction model which includes two types of controller which is either controlled by the keypad or head movement.

4.1 Implementation of MARMOT

MARMOT is implemented based on the software requirement documentation of the intelligent wheelchair. The discussion of the modeling for intelligent wheelchair will be divided into two separation descriptions which are specification and realization. As mentioned before, specification will produce three types of model which are functional, behavior and structural model. It is represented by operation specification table, UML state diagram and class diagram respectively. Meanwhile, realization will produce activity and structural model and interaction model. It is represented by activity diagram, class diagram (see Figure 4) and sequence diagram (see Figure 5) respectively. The difference between specification and realization class diagram is the detailed information where specification class diagram just includes basic information. On the other hand, realization class diagram will provide detailed information such as the operation and the attribute.

Figure 2. The sensors and actuators in the intelligent wheelchair

Figure 3. The intelligent wheelchair use-case diagram

Figure 4. Class diagram-sensor (realization)
4.2 Implementation of Hybrid Approaches

Intelligent wheelchair containment hierarchy as shown in Figure 6 includes the hardware components and software components. Hardware component represented by stereotype «<<hardware>>» and it has six hardware components such as IRSensor, sonar sensor, controller, keypad, motor and accelerometer. Controller component will interact with intelligent wheelchair application using the device driver. In this stage, the hybrid approaches are implemented where the hierarchy of the intelligent wheelchair system which does not only include the component name but also the detailed information about the input and output is shown. Therefore, this information can easily help ERT software developer to compose component application as described at the following Subsection.

4.3 Implementation of PECOS

In this stage, component composition is fully using PECOS concept where the interface of the component is represented by ports. To represent active component, time and priority initially will be set in the right site of the component. If there is no setting value for timing and priority in that component, it means that the component is a passive component. However, for this time, an event component type is not considered. Component-Oriented Programming (COP) framework is used to design the component composition. A new COP framework was proposed to better support embedded resource-constrained ERT graphical block components integration and composition based on PECOS model [11]. The COP programming framework was aimed towards visual programming paradigm for ERT programming based on component-base software engineering approach. Figure 7 illustrates the intelligent wheelchair system which includes eleven active components and seven passive components. Besides, it consists of eight leaf components (without sub-component) and nine composite components (with sub-component) shown by blocks with shadow. Very single component provides the ports and connection line to show the composition of the overall intelligent wheelchair.

Lastly, the following code shows in Figure 8 a fragment of the configuration definitions in the header file for the intelligent Wheelchair component.
5 COP TOOL DESIGN

A Class diagram for the overall tool includes four packages, five classes and their respectively relationship. New component package includes seven classes which are component aggregate class with active component, passive component and class port where it inherits with class inport and ouport and associate with constant class. New component package also have relationship with other class diagram and package which is component association class relationship with component list class and GUI ERT developer, compose relationship with database package and realization relationship with connection rule class.

This tool will be divided into three modules. The first module will be supported by package NewComponent which is required to develop new component including component name, port type, component type and allow constant values. The second module will be supported by NewComponent and ComponentConnection packages. Component connection package includes two class diagram which are connector and connection rule. Connector is used to integrate one component to other components and at the same time it also must fulfill several rules that were mentioned before. Component may constitute existing component that can be retrieved from component list at ComponentRepository from the database or develop new component if necessary. The third module is supported by File class diagram, UserGuide class diagram and package ManageBar. In File class diagram, it includes five operations which are Save() is to store all the work doing at workspace into some
directory or database, Open() is to open the existing work that is already saved at directory or database, Print() is to print the current work or previous work after saving. Exit() is to close from the current workspace and SaveAs() is to save the current or existing work into others directory.

To develop these tools, Visual C# is used. This tool only supports design and implementation phase whereby analysis phase will model using UML 2.0. The purpose of this tool development is to support multi-constraint problems at design and implementation phase to ERT software development where knowledge from PECOS component infrastructure is used. Multi-disciplinary problem will supported at analysis phase using MARMOT method knowledge.

However, this Section only focuses on multi-constraint problems that is supported at design and implementation phase. The overall use case of the tool is shown in Figure 9. In this tool, it includes three main modules which are the development component, integrate component and manage file. For the first module, it includes three activities that are resized component, insert port and set properties. From the use case diagram, second module will be supported by integrate component use case in which it includes two use cases which are component composition and component connection. The third module will be supported by manage file and it includes managing view file and request help.

5.1 User Interface

Figure 10 shows the prototype first version of the COP tool. It includes seven main menus, which are File, View, Component Development, Component List, Component Integration and Help.

File menu includes several operation which are Save(), Save As(), Open(), Print() and Exit(). View menu includes two others sub-menus which are the view toolbar and view status bar. For the component development menus, it includes four sub-menus that are new component sub-menu, set properties, set constant and color sub-menus. Help sub-menu is to provide user guidance to guide user how to use this tool and some guidance to solve problems that often occur.

New component is to put components block diagram into workspace. Whereby set properties sub-menu is to set component name. If component type is active, timing and priority will be set together. However, if component type is passive, it is not necessary to put any data in the timing and priority box.

A constant value allows input data in the component block diagram. Therefore, constant value will be set in set constant sub-menu. In the setting to put constant value, constant parameter needs to set how much input is necessary in particular component block diagram. After setting the number of parameter needed, it has to put the parameter name, value and data type.

Component list sub-menu is to get available existing component that already develop in repository. User can reuse the existing component to develop new application.

Currently, only the first module is completed. While this tool can only develop new component, change workspace background color, set properties and set constant values. The other menus listed above are in progress of the tool development phases.
6 CONCLUSION

The hybrid approaches of the component technologies aims to support multi-disciplinary knowledge and multi-constraint requirement of CBD for ERT software development. From the application of the approach in an intelligent wheelchair case-study, it was demonstrated that the proposed hybrid approaches has helped to guide the developer of ERT software from analysis of the multi-disciplinary knowledge to design the multi-constraint extra-functionality until the implementation of the composition and generation code template using component-oriented programming (COP) frameworks. The hybrid approaches also able to support component-based methodological in three methods: component-oriented analysis (COA), component-oriented design (COD) and COP. Prototype of the COP tool focused only on graphical component development that includes creating new component, setting properties, setting constant value and changing background of the workspace. Set properties include two activities, which are set timing and priority to support multi-constraint as needed in ERT software development. This tool will extract analysis models from MARMOT method designed using UML 2.0. The advantages of these tools are supported specification and component repository in graphical block form. Besides that, it is more useful and effective to support methodology or framework. Therefore, this tool will help analysts and designers to conduct the various activities within the CBSE productively. For future works, an extension of UML 2.0 will be adapted into the COP tool.

ACKNOWLEDGEMENT

Thank you to the Ministry of Science, Technology and Innovation (MOSTI) Science Fund for the financing and funding, Universiti Teknologi Malaysia (UTM) for the facilities and infrastructures and our Software Engineering Lab 2 members for their continuous support.

REFERENCES