5

A Robotic Wheelchair Component-Based Software Development

Dayang N. A. Jawawi et al.*
Universiti Teknologi Malaysia
Malaysia

1. Introduction

A robotic wheelchair system provides mobility for handicapped and elderly people who are unable to operate classical wheelchair system. Software development for such system is challenged by requirement for multi-disciplines expert knowledge which includes embedded systems, real-time software issues, control theories and artificial intelligence aspects. Software reuse is an approach to provide a way to reuse expertise that can be used across domains in software engineering. Software reuse can be a mechanism to support the attempts to transfer technology from other engineering fields to rehabilitation engineering. For example, (Bonail et al., 2009) and (Cheein et al., 2009) have attempted to transfer software platform and algorithms from robotic technologies to rehabilitation engineering software development. The technologies transfer requires a methodological support to enable a systematic software reuse of the multi-disciplines expert knowledge.

Software reuse is one of the promising approaches to increase software productivity and improve its quality, as well as to decrease the costs of software development. This is because of software reuse uses existing software either in the form of component or knowledge to construct new software. Yet, applying software reuse in Embedded Real-Time (ERT) domain, such as robotic wheelchair sets major challenges to the software development process due to the resource-constrained and real-time requirements of the system.

In order to overcome multi-constraints and multi-disciplinary knowledge in ERT software development problems, Component-Based Development (CBD) method becomes a promising approach for ERT software development (Bunse & Gross, 2006; Carlson et al., 2006). Existing industrial component 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 non-functional properties in ERT systems.

With the purpose to meet the requirements of ERT systems, a number of component technologies such as Koala (Ommering, 2000), PECOS (Nierstrasz, et al., 2002) and KobrA (Atkinson, et al., 2002) have emerged. However, these component technologies still have some weaknesses. Koala and PECOS cannot support multi-disciplinary knowledge, but they can

* Suzila Sabil, Rosbi Mamat, Mohd Zulkifli Mohd Zaki, Mahmood Aghajani Siroos Talab, Radziah Mohamad, Norazian M. Hamdan and Khadijah Kamal
Universiti Teknologi Malaysia, Malaysia

support different multi-constraint extra-functionality requirement. On the contrary, Koala supports resource constraint and PECOS supports timing problems. KobrA does not support ERT system development, but it has an extension that calls MARMOT (Bunse & Gross, 2006) to support multi-disciplinary knowledge in ERT system development. The only limitation of MARMOT is minimum support for multi-constraint extra-functionality requirement.

Therefore, PECOS is suitable to support multi-constraint in extra-functionality requirements whereby, MARMOT is good to support multi-disciplinary knowledge. An integration of PECOS and MARMOT can be a promising strategy to support the two issues. Implying a methodological support to enable a systematic CBD can be an important approach with consideration of the two issues. Currently, there are many Object-Oriented Analysis and Design (OOAD) methodology available, but for Component-Oriented Analysis and Design (COAD) method, the focus is still on PC based domain such as the web application (Lee and Shirani, 2004) and simulation systems (Gong et al., 2010). Component technologies also improved along with engineering practices, but they lack of a methodology that uses components within such a paradigm (Dogru & Tanik, 2003).

Motivated by these challenges, the focus of this chapter is to propose a method for developing robotic wheelchair software using a set of reusable software components obtained from mobile robot software. The method was adapted from general ERT component technologies and was applied to a robotic wheelchair CBD. The method is a combination of MARMOT and PECOS technologies aiming to support CBD methodological with purpose to solve multi-constraint extra-functionality requirement and multi-disciplinary knowledge that are required in the robotic wheelchair software development. The proposed systematic CBD process model is based on the Component-Based Software Engineering (CBSE) that is defined by Wang and Qian (2005). CBSE is a combination of Component-Oriented Analysis (COA), Component-Oriented Design (COD) and Component-Oriented Programming (COP) and Component-Oriented Management (COM). This chapter focuses on COA, COD and COP process of development, and this chapter defines the COA, COD and COP modelling and deployment activities of the method in a form of process model representation. The process model depicts understandable integration between MARMOT and PECOS.

This chapter documented the applicability of the process model in a wheelchair software development and implementation. This implementation showed how the process model helped the wheelchair hardware and software engineer to identify the possible software reused component in the early stage of the system and software development. The amount of software component reused in the wheelchair CBD from a mobile robot system was discussed and compared with a reused case from a mobile robot to another mobile robot CBD. The objective of the comparison was to identify the differences between the software reuse to support technologies transfer from robotic technologies to rehabilitation engineering with software reuse within robotics systems. Software reuse in robotics domain is one of the focuses area in current robotic research, example of the study are by Nesnas et al. (2006), Jang et al (2010) and Mallet et al. (2007).

The layout of this chapter is as follows: Section 2 discusses the robotic wheelchair design considerations and the hardware description. In section 3 the strategy to define the process model to support the CBD of the wheelchair software are described. The process model to support the CBD method is described in Section 4. Section 5 illustrates the process model in a CBD of the robotic wheelchair software. The comparison result to compare the process model with other models will be discussed in detail in Section 6. The Section 7 concludes the chapter.

2. The robotic wheelchair system

A prototype of robotic wheelchair was developed to support our researches in ERT software engineering and rehabilitation robotics. The main considerations when developing this prototype were to have a low cost and 'open' system such that it enables different aspects of hardware and software experimentations to be performed.

Due to these considerations, rather than basing the prototype on a standard power wheelchair such as the smart wheelchair system developed by Simpson et al. (2004), a commercially available manual wheelchair was used as the base for the robotic wheelchair prototype. An easily detachable add-on unit was developed to be attached to the manual wheelchair without significant modifications to the manual wheelchair. This add on unit consists of two geared direct current (DC) motors, two 12V batteries and a control box with embedded processor and associated electronics. Fig. 1 shows the prototype of robotic wheelchair which consists of a standard manual wheelchair and add on unit. The DC motors together with two small wheels provide the motorized wheels for mobility. The DC motors are powered by the 24V 14 Ampere obtained from the two 12V batteries. The voltage regulators in the control box provide 5V supplies for the sensors and the embedded processor from the two batteries.

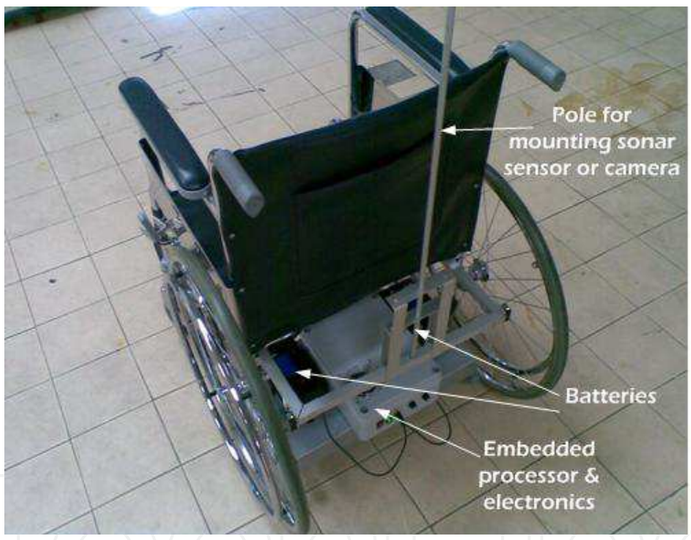


Fig. 1. The prototype of robotic wheelchair. The motorized wheels are under the add on unit

To provide the capability for sensing the environment, sets of sensors are attached to the robotic wheelchair. Infra red (IR) distance sensors are used to detect obstacles in the direction of the robotic wheelchair movement. Four IR sensors are mounted at the front and two IR sensors are mounted at the back. Sonar sensor is used to map room environment or detect far obstacles. It is mounted on the pole above the user head.

Currently, the robotic wheelchair supports two ways of controlling. The first way is through the usual joystick or keypad control. The second way is through the movements of head. Head movements control is particularly useful for severely-handicapped people who have spinal cord injury or quadriplegia which cannot use their hands to control the wheelchair. In the developed prototype, the head movement control of the robotic wheelchair is achieved with the help of accelerometer or tilt sensor. The accelerometer senses head movements and based

on the predetermined direction of movements, the robotic wheelchair can be controlled accordingly. Fig. 2 shows the side view of the robotic wheelchair with the locations of sensors.

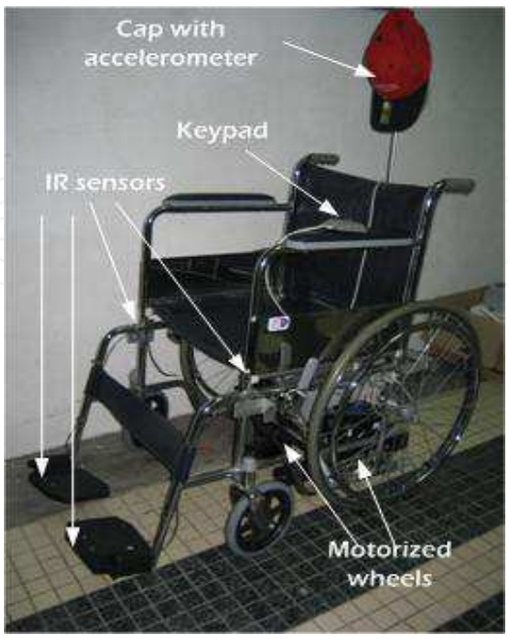


Fig. 2. The side view of robotic wheelchair prototype with sensors locations.

The embedded processor provides the intelligent decision making and motor control. All the sensors outputs are processed by the embedded processor and control the two DC motors as desired by the user. Automatic reactions such as stop or avoid when obstacles are detected are also handled by the embedded processor.
In the robotic wheelchair prototype an ATMEL ATMega32 8-bit single-chip microcontroller is used as the embedded processor. The ATMega32 has 32 Kbytes of flash program memory, and 2 Kbytes internal Random Access Memory (RAM). The ATMega32 also includes an 8-channels 10-bit analogue-to-digital converter (ADC), three timers, parallel input-output ports and several serial communication interfaces including Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C) and Universal Synchronous Asynchronous Receiver Transmitter (USART). Fig. 3 shows the interfaces between the embedded processor, sensors and actuators in the robotic wheelchair.

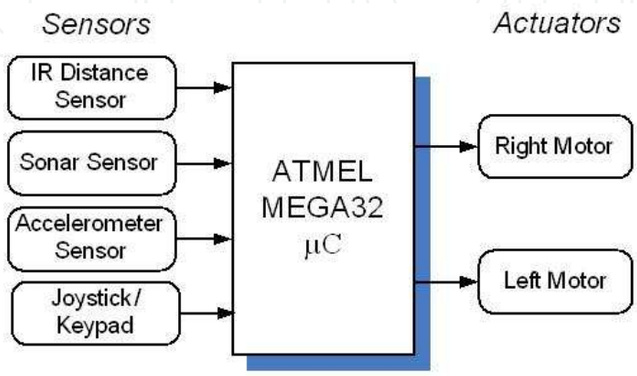


Fig. 3. The sensors and actuators in the robotic wheelchair.

3. The strategy to define the component-based development phases

A strategy is needed to allow the mapping of MARMOT and PECOS methods in order to identify overlapping phases in between the two methods. The mapping of these two methods is required in order to propose a new component-oriented developed method based on the two selected component models. Fig. 4 shows the models and phases of the methods. It comprises of three development process which are COA, COD and COP. The development process is then divided into five phases, which are analysis, early design, detailed design, composition and implementation. COA level involves the analysis phase; COD level involves two phases, which are early design and detailed design; and COP level involves composition and implementation phase.

The analysis phase includes two strategies, which are preliminary information and system architecture planning. Preliminary information consists of the basic requirements of the use case diagram, use case description and interaction model. The system architecture planning transforms the circuit into Unified Modelling Language (UML) representation and in turn produces a preliminary containment hierarchy of the whole system.

The early design phase contains the finer-grain component of software that further divides into two parts, which are specification and realization. In specification of components, three models are produced, which are functional model, behavioural model, and structural model. The structural model is also produced in realization of components, as well as the activity model and interaction model.

<table>
<tr><th>LEVEL</th><th>CBD PROCESS</th><th colspan="2">MODEL DIAGRAM</th><th>CM</th></tr>
<tr><td rowspan="4">COA</td><td rowspan="4">Analysis Phase</td><td colspan="2">Preliminary Information</td><td rowspan="9">MARMOT Method</td></tr>
<tr><td colspan="2">- Use Case Diagram
- Use Case Description
- Interaction model</td></tr>
<tr><td colspan="2">System Architecture Planning</td></tr>
<tr><td colspan="2">- Hardware UML representation
- Preliminary Containment Hierarchy</td></tr>
<tr><td rowspan="5">COD</td><td rowspan="3">Early Design Phase</td><td colspan="2">Finer-Grain Component</td></tr>
<tr><td>Specification
- Functional Model
- Behavioural Model
- Structural Model</td><td>Realization
- Structural Model
- Activity Model
- Interaction Model</td></tr>
<tr><td rowspan="2">Detailed Design Phase</td><td colspan="2">Details Information</td><td rowspan="2">Integration Point</td></tr>
<tr><td colspan="2">Regular Containment Hierarchy</td></tr>
<tr><td rowspan="4">COP</td><td rowspan="2">Composition Phase</td><td colspan="2">Integration Process</td><td rowspan="4">PECOS Component Model</td></tr>
<tr><td colspan="2">Composition Diagram</td></tr>
<tr><td rowspan="2">Implementation Phase</td><td colspan="2">Generation Code</td></tr>
<tr><td colspan="2">Code Template</td></tr>
</table>

Fig. 4. Models and phases of the method

In order to elaborate upon the preliminary hierarchy containment based on the finer-grain component, regular containment hierarchy is produced at the detailed design phase. Thus at implementation phase, the integration process is performed, and it is represented by the component composition diagram. Based on the component composition diagram, a code template is generated, which includes single component code template and component composition code template.

The models and phases in this research are derived from MARMOT and PECOS models to create a new method; the MARMOT method is applied at analysis, early design and detailed design phase while PECOS is applied at detailed design, composition and implementation phase. The integration point between MARMOT method and PECOS component model is at the detailed design phase, as shown in Fig. 4.

The central notion in COA and COD is the component. COP supports constructing software systems by composing independent components into software architecture. Software component is grouped with its component infrastructure. Component infrastructures include three elements, which are component model, component connection and component deployment. The analysis phase supports COA method, whereas early design and detailed design phases support COD. COP method is supported by two phases, which are composition and implementation phase. The next section provides detailed discussion on the process model of the integrated MARMOT and PECOS method. The process model clarifies the integration point of the two methods.

4. A component-based development process model for embedded real time software

Defining the integrated process of MARMOT and PECOS methods in a systematic form is important to enable and support the development of CASE tool for ERT system. Here, the process is represented using Software Process Engineering Meta-model (SPEM) (Schuppenies & Steinhauer, 2002). SPEM is a meta-model that is used to describe a concrete software development process or family of related software development process. The Fig. 5 below shows the SPEM icons that are used in this project:

a. Activity: is the main subclass of Work Definition, it describes a piece of work performed by one Process Role.
b. Document: a stereotype of work product.
c. Process Role: the performer of Activities and responsible for a set of Work Products.
d. Phase: a specialization of Work Definition such that its precondition defines the phase entry criteria and its goal (often called a "milestone") defines the phase exit criteria.
e. Work Definition: kind of Operation that describes the work performed in the process.
f. Work Product: an artifact is anything produced, consumed, or modified by a process.

There are five phases in the process model; analysis phase, early design phase, detail design phase, composition phase and implementation phase. Each of these phases produces a work product for the related activities and has its own responsibilities in designing one or more artefacts. Fig. 6 illustrates the phases in COA, COD and COP process using SPEM. The details for each phase are discussed in the following subsection.

Fig. 7 illustrates the use case diagram for analysis phase, whereas Fig. 8 represents the use case for early design and detailed design phase. Meanwhile, Fig. 9 illustrates the use case diagram for composition and implementation phase. These use case diagrams show the

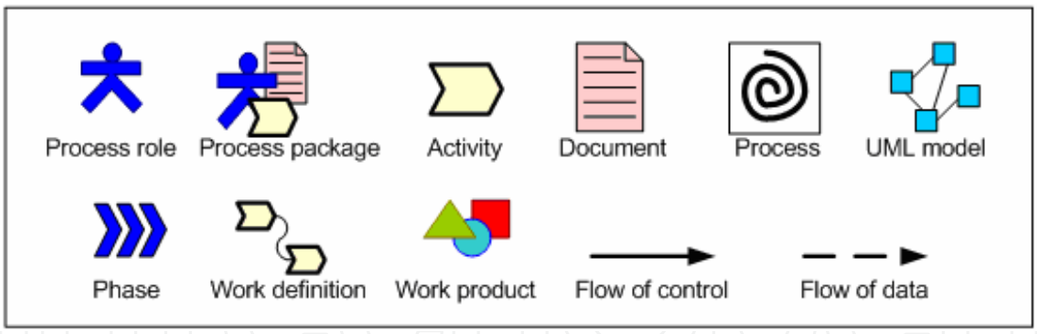


Fig. 5. SPEM Icons

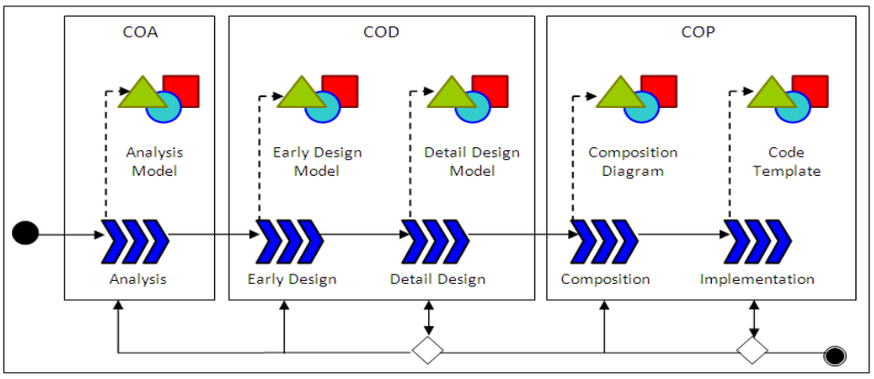


Fig. 6. Process model phases for the method

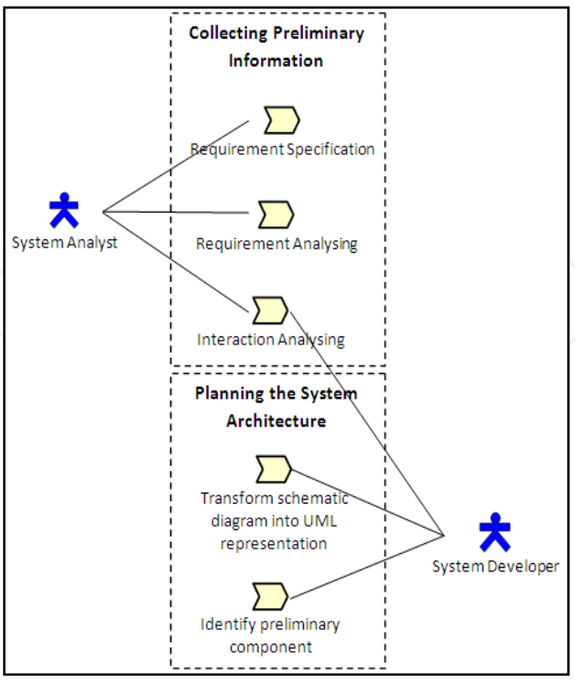


Fig. 7. Use case diagram for analysis phase

relationships between the process and the main activities in the software process based on the three level developments, which are COA, COD and COP.
In the COA level, analysis phase is divided into two main activities. The first activity is collecting preliminary information which consists of three processes; requirement specification, requirement analyzing and interaction analyzing. The second activity is planning the system architecture which includes two processes; manually transforming electrical schematic diagram into UML representation and identifying the preliminary component.

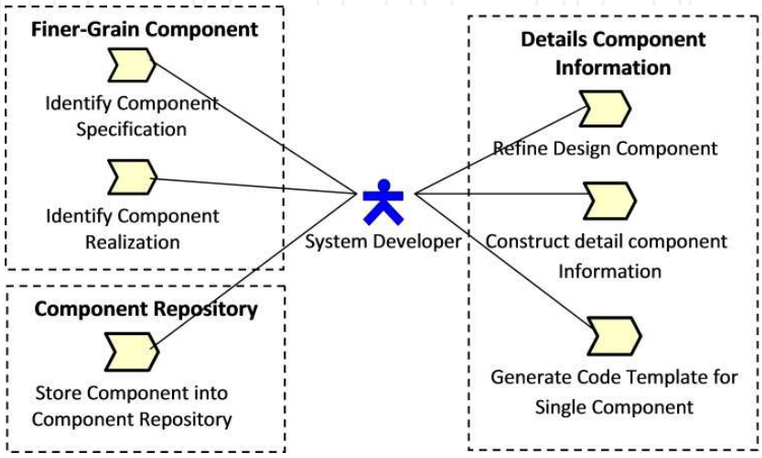


Fig. 8. Use case diagram for early design and detailed design phase

The COD level includes two phases, which are early design and detailed design phase. In the early design phase, the finer-grain component includes two activities. The first activity is identifying component specification that will produce a functional model, behavioural model and structural model. The second activity identifies component realization that

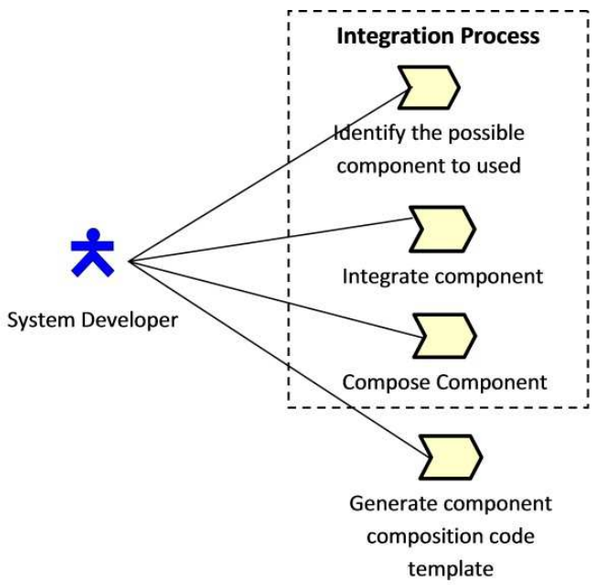


Fig. 9. Use case diagram for composition and implementation phase

produces a structural model, activity model and interaction model. 1 Meanwhile, in detailed design the component information is transformed into a graphical representation. The activities are refining design component, constructing detail component information and generating the code template for single component.
The COP level of CBD process consists of composition and implementation phase. Process integration starts at the composition phase and includes three activities, which are identifying the possible component, component integration and component composition. The implementation phase is to generate component composition code template.

4.1 Analysis phase

The purpose of analysis phase is to analyze the ERT system requirement. As was previously mentioned, ERT system requirement involves multi-disciplinary knowledge. Therefore, analyzing the ERT system requirement does not focus on software only but also on the hardware. In this analysis phase, the MARMOT method is used to support the multi-disciplinary knowledge required for ERT system. It is divided into two parts; the first part is collecting the preliminary information and the second part is planning the system architecture. Collection of the preliminary information can be further divided into three activities, which are problem analysis, requirement description and interaction analysis. System architecture planning includes two activities, which are transforming electrical schematic diagram and identifying a preliminary component based on transformation electrical schematic diagram into UML representation.
The analysis phase starts with analyzing the requirement, by identifying the problems and by looking at the application functionalities. It also identifies the user who is interacting with the application. This process produces the use case model of the system. The next activity involved is describing the requirement description thoroughly so as to produce the use case description table. This description table describes the multi-disciplinary knowledge of ERT system requirements. After that, interaction analysis is carried out to briefly draft the interaction between each state in order to produce the interaction model. Fig. 10 shows the flow of the activity in analysis phase, which is described in terms of work definition and work product as input.
The next step involves in obtaining the information directly from the electrical schematic diagram in order to identify the possible preliminary component software. The electrical schematic diagram is manually transformed into UML and all hardware parts are removed to obtain a list consisting of only software parts. The software parts can be a component with a stereotype <>. The list of the possible software components is represented by the containment hierarchy diagram. Fig. 10 shows the work definition and work product of analysis phase.

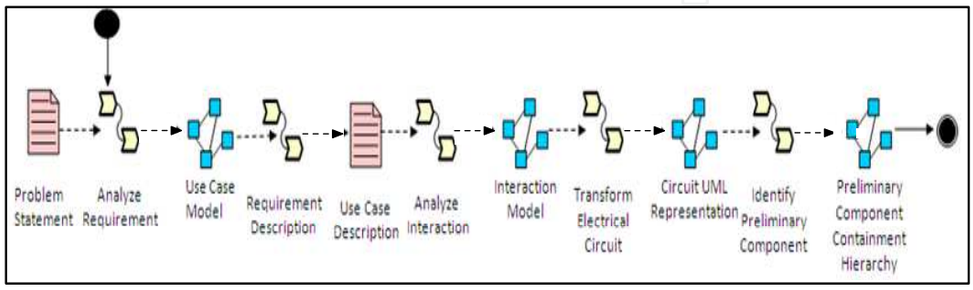


Fig. 10. The analysis phase description in terms of work definition and work product

4.2 Early design and detailed design phase

The purpose of an early design phase is to fine-grain the preliminary component containment hierarchy that is identified from the analysis phase. The preliminary component containment hierarchy at analysis phase is used as input for early design phase.

The early design phase has two main activities, which are identifying component specification and component realization for each software component. These two activities produce UML diagram artefacts. Component specification produces three models, which are structural, behaviour and functional model whereas component realization also produces three models, which are structural, activity and interaction model. In this stage, multi-constraint extra-functionality requirement which focuses on timing is considered as model for the software component specification and realization, and is represented by timing diagram.

At the detailed design phase, the integration point between the MARMOT and PECOS concept is produced. The detailed design phase includes three activities, which are refining the design component, constructing detail component information, generating the code template for single component and storing component into the component repository. Refinement of the design component involves some additional component to model the software behaviour. As a result, it produced a new version of preliminary component containment hierarchy. The process repeats until the component containment hierarchy fulfils the multi-disciplinary requirement. After the preliminary component containment hierarchy matures, the detailed component information is constructed.

In this phase, PECOS modification is integrated whereby each component includes not only the functional requirement but also multi-constraint extra-functionality requirement. In this research, the focus is on timing of extra-functionality requirement. Therefore, regular component containment hierarchy is produced where the component includes timing, priority, outport and inport. Based on the detailed information of each component, the code template produces and stores the component into the component repository. It then produces component lists of the entire application. Fig. 11 shows the steps at early design and detail design phase.

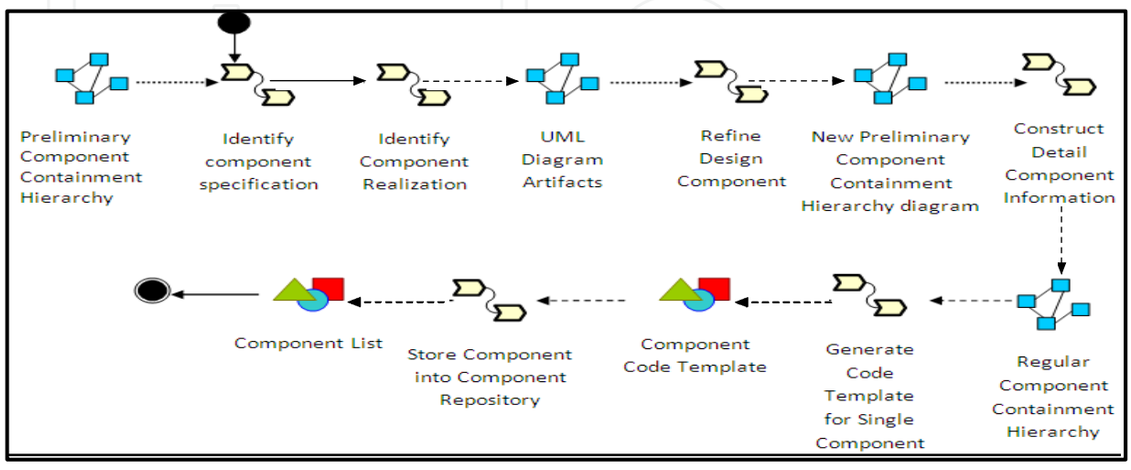


Fig. 11. Design phase description in terms of work definition and work product

4.3 Composition and implementation phase

The purpose of this phase is to integrate the components and generate the composition component code template for the entire application. The integration process integrates more than one component. In this stage, a component can have a sub-component, and it produces the composition component diagrams for the application.

Once the composition is completed, the next step is to allocate property bundles value such as period and priority for active components using real-time scheduling theory. Meanwhile, the code template for the composition diagram is generated in the implementation phase. Fig. 12 illustrates each step at composition and implementation phase.

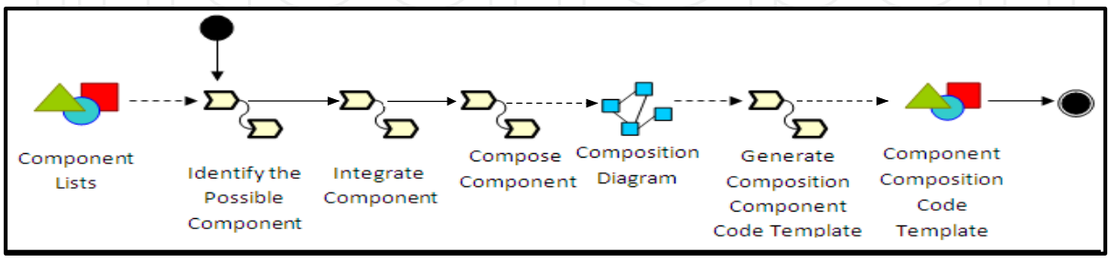


Fig. 12. Composition and implementation phase description in terms of work definition and work product

5. The wheelchair software development

The Intelligent Wheelchair (I-Wheelchair) case study, described in Section 2, was implemented to apply the integrated process model and show how the reuse activities from mobile robot systems to the I-Wheelchair were performed. The I-Wheelchair case study is an embedded system and it is relatable with the resource constraint of real-time and multi-disciplinary requirements.

I-Wheelchair software development as represented by the process model includes five different phases, which are analysis, early design, detailed design, composition and implementation phase. The following sub-section further elaborates each phase.

5.1 Analysis phase

As mentioned before, MARMOT method is implemented at the analysis phase. In this case, this phase involves processes like defining diagrams and textual specifications from the context realization of the I-Wheelchair system. It is divided into two parts, which are preliminary information and system architecture planning. Preliminary information produces use case diagrams, use case description and interaction model; and system architecture planning produces hardware UML diagram and preliminary containment hierarchy. The following sub-sections will be based on the I-Wheelchair case study.

5.1.1 Preliminary information

Preliminary information includes three activities which are requirement specification, analysis requirement and analysis interaction. The requirement specification of I-Wheelchair system is represented by the use case diagram, in which it consists of a textual and a graphical representation as shown in Fig. 13.

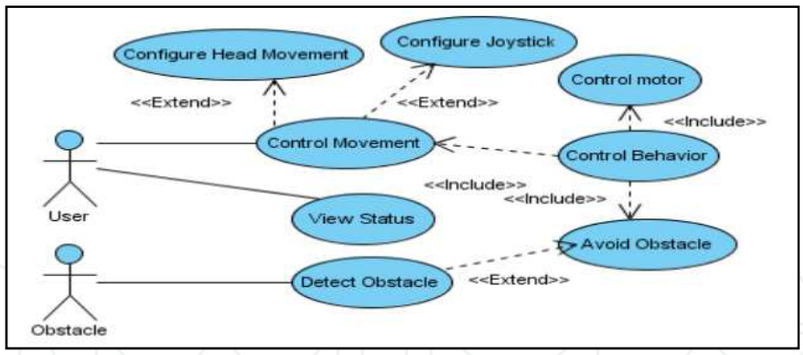


Fig. 13. I-Wheelchair use case diagram

The actor User initiates the task of controlling the I-Wheelchair movement whereby the actor Obstacle initiates the task of avoiding an obstacle around its environment. Control Movement and View Status use cases is controlled by User actor. Meanwhile, Detect Obstacle use case is controlled by Obstacle actor and extended by Avoid Obstacle use case. The use case description table is used to provide more information with regards to the I-Wheelchair use case diagram. An example of a detailed use case description table for Detect Obstacle use case is as shown in Table 1. The description table provided detailed information of Detect Obstacle use case which includes the name of use case, responsible actors, use case goals, use case descriptions, exceptions, rules, quality requirements, input/output, pre and post-conditions of the use case.

Name	Detect Obstacle
Actor	Obstacle
Goal	To detect any obstacle at front or back using IR sensor
Description	Get input data from sensor (HMC) or signal (joystick) to detect obstacle at the front or back in range distance min=30 cm and max=80 cm.
Exception	N/A
Rules	N/A
Quality Requirement	N/A
Input/Output	Input: IR sensor back IR sensor front Output: Motor
Pre-Conditions	Detect any objects that defend the movement whether at the front or back
Post-Condition	Move follow current direction

Table 1. Detect Obstacle Use Case Description

To analyze the interaction of the I-Wheelchair system, the interaction model diagram is used as shown in Fig. 14. The purpose of an interaction model is to represent the general flow of the I-Wheelchair control movement and obstacles avoidance by the two actors; User and Obstacle. It also illustrates several alternative actions that can be performed, and represents typical interaction of the operation for the overall system.

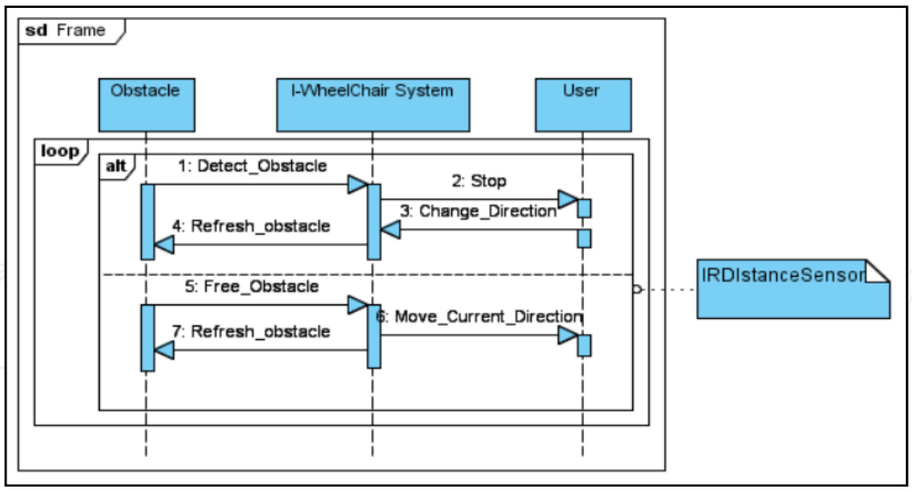


Fig. 14. Obstacle interaction model for avoiding obstacles

Fig. 14 shows the interaction model for Obstacle use case. This model represents behaviour of the Obstacle use case, in which includes two behaviour: if the I-Wheelchair system detects an obstacle, it will stop moving and change the direction of its movement, and after that it will start to detect other obstacles; if no obstacle is detected, the system will move according to the current direction that is given by the user and it will start to detect other obstacles. A semaphore used in the implementation of the Obstacle interaction model to avoid deadlock situation.

The infrared distance sensors at the back and front of the I-Wheelchair system are used to detect obstacles that labelled as input to the system as mentioned in Table 1. Therefore, the use case diagram, use case description table and interaction model can be useful for the ERT system developer to translate the hardware requirement into software requirement at the analysis phase.

5.1.2 System architecture planning

In this stage, the system architecture is planned by identifying the preliminary components. The identification of the component hardware is done based on the electrical microcontroller schematic diagram. As it requires software to calculate the control signal to be sent to motors, the motor is considered one of the initial hardware components. The microcontroller schematic diagram will be transformed into the UML presentation manually. The UML representation diagram represents the hardware component that may include software component.

Therefore, the five hardware components identified are Sensor, Controller, Joystick, Motor and Human Computer/Robot Interaction (HCI or HRI). These components are represented by a component tree called preliminary containment hierarchy as shown in Fig. 15. The I-Wheelchair is composed of both hardware and software components. The Controller hardware component consists of Driver and Application, with the Driver acting as the communicator between the software and the hardware components or called "wrapper hardware" in MARMOT. Hence, software component should be under the Driver component and an Application component refers to the behaviour of the software component.

5.2 Early design phase

MARMOT was implemented based on the software requirement documentation of the I-Wheelchair. The discussion on the modelling for the I-Wheelchair was divided into two separate descriptions; specification and realization. As mentioned before, specification produced three types of model; functional, behavioural and structural model, represented by the operation specification table, UML state diagram and class diagram respectively. Realization produced the activity and structural model as well as the interaction model which are represented by the activity diagram, class diagram and interaction diagram respectively. The difference between the specification and the realization class diagrams are the details of the information provided where the specification class diagram only included the basic information while the realization class diagrams provided more detailed information such as the operation and the attribute. Fig. 16 and Fig. 17 show the examples of realization models, i.e. the Sensor class diagram and the Sensor interaction diagram.

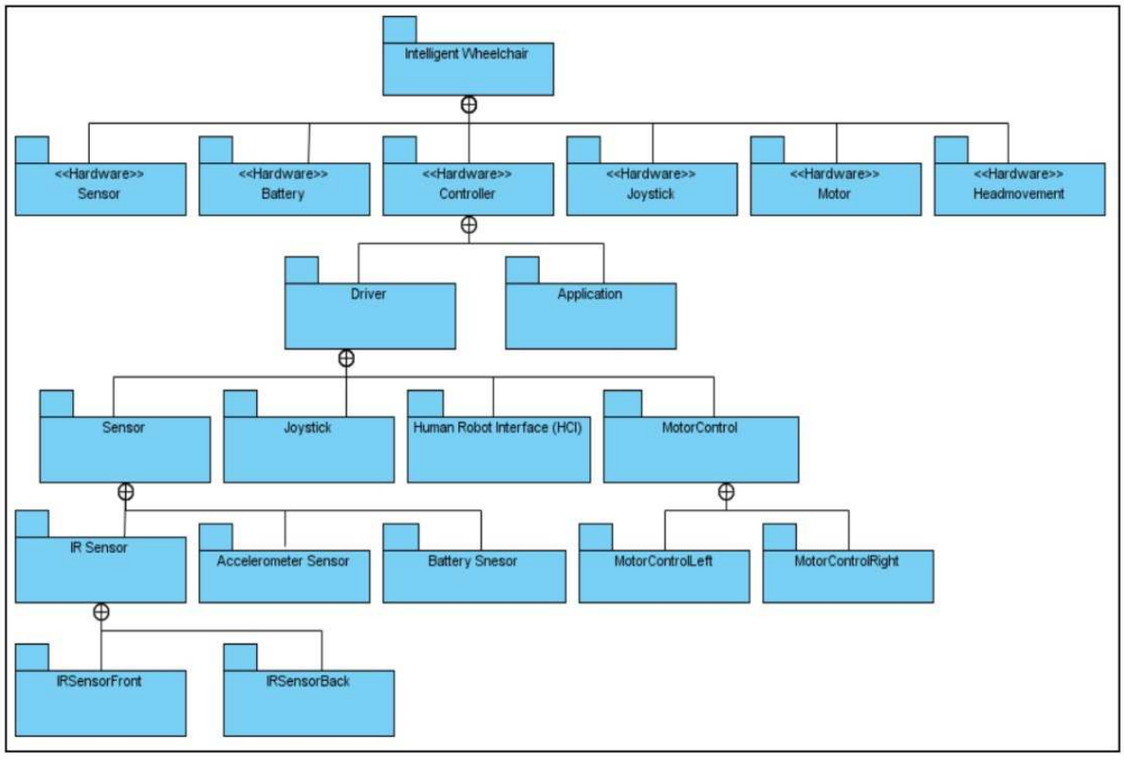


Fig. 15. Preliminary Containment Hierarchy

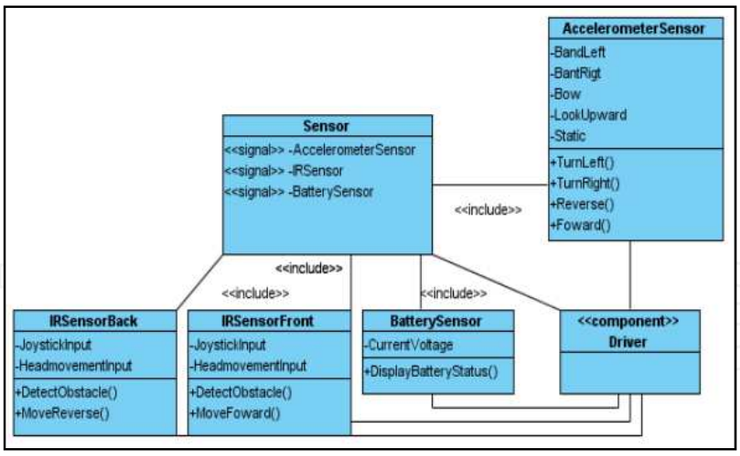


Fig. 16. Structural Model (Class Diagram)-Realization (Sensor)

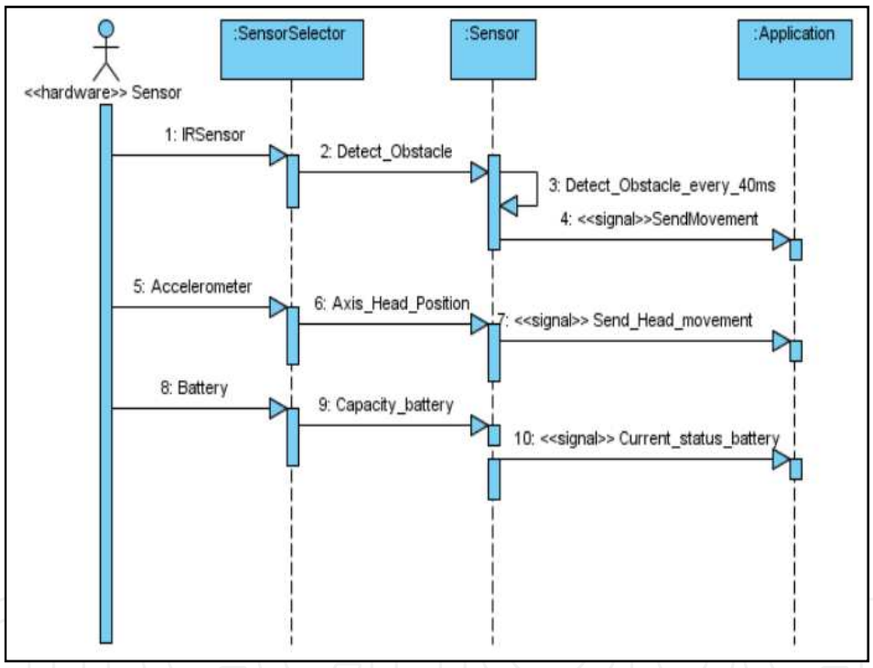


Fig. 17. Interaction Model-Realization (Sensor)

5.3 Detailed design phase

The integration of MARMOT and PECOS was implemented in this phase by extending the MARMOT Regular Containment Hierarchy diagram. The hierarchy diagram of the I-Wheelchair system in this phase is derived and extended from the Preliminary Containment Hierarchy diagram from Fig. 15, which included the component name and also has the detailed description of the component diagram information. The regular containment hierarchy of the I-Wheelchair shown in Fig. 18 includes the hardware and software components. The hardware component is represented by the <component> stereotype with six hardware components such as sensor, LED, controller, joystick, motor and accelerometer

and was initially derived from the schematic diagram. The Controller component interacts with the I-Wheelchair application using the device driver to support software component structural modelling.

The component definition used in this work was extended from the original PECOS model (Jawawi et al., 2006). Fig. 19 shows an example of a composite component definition. The component diagram also contained component types such as active and passive. In order to represent an active component, initial time and priority are set in the right side of the component. If there is no set value for timing and priority in that component, it means that the component is a passive component.

Furthermore, each component has port information which consisted of two types of port; inport and outport. The port name is represented by a code such as IP00, signifying that the component has an inport port at the first position followed by IP01 for the inport port at the second position as illustrated in Fig. 19. The same rules were applied for the outport port. The components without shadow represent a leaf component (without any sub-components inside) while the components with shadow indicate that it has sub-components called composite components as illustrated in Fig. 18.

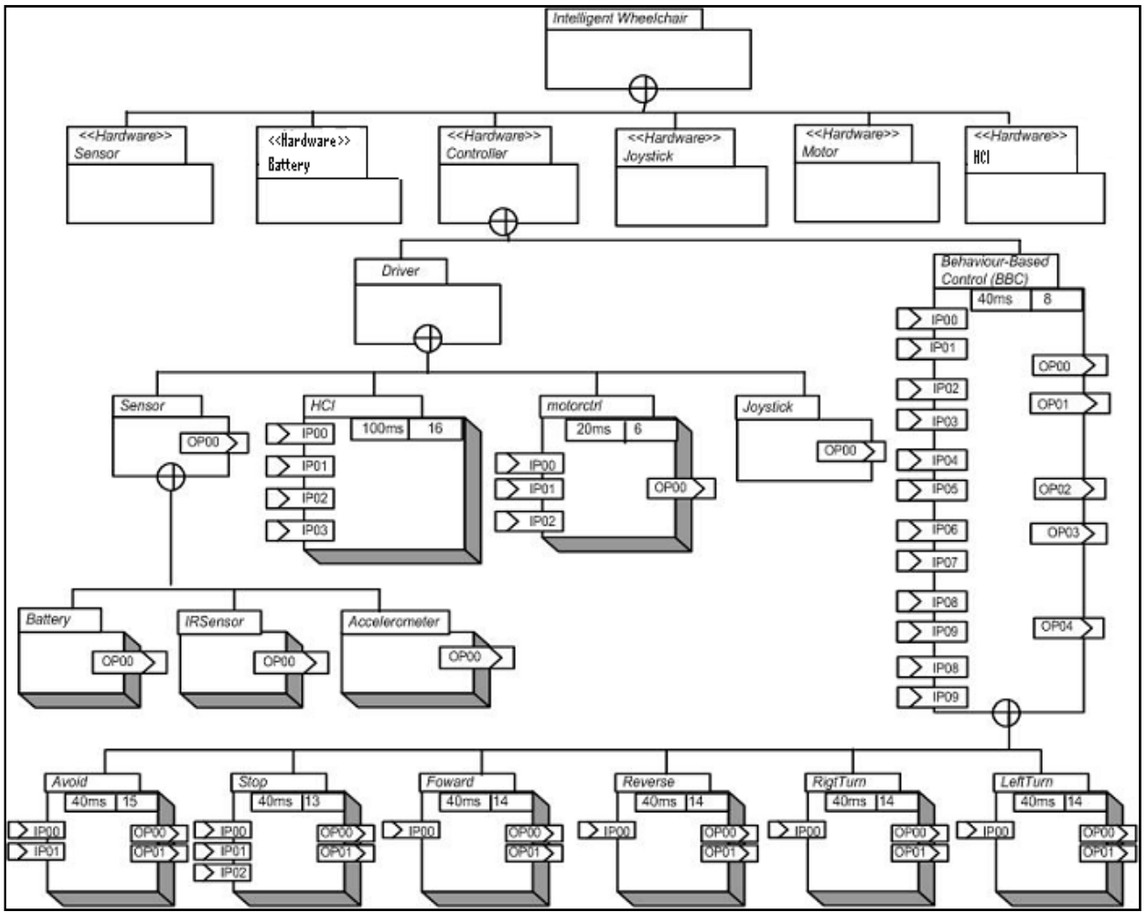


Fig. 18. Regular Containment Hierarchy

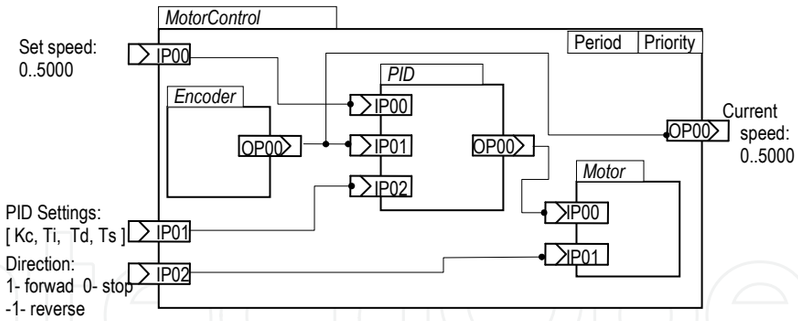


Fig. 19. MotorControl composite component

The Driver component is made up of four software components which are Sensor, HCI, Motorctrl and Joystick as shown in Fig. 18. These four software components were initially derived from hardware components where some modifications have been made based on the software design; for example, the Battery hardware component was not utilized as a Battery software component because Battery is one of the sensors.

The purpose of the Application software component is to support behaviour modelling of the I-Wheelchair application. Behaviour Based Contol (BBC) was used in the I-Wheelchair application to support behaviour modelling where BBC controls the behaviour of the I-Wheelchair with its six different behaviours including AvoidObstacle, Stop, Forward, Reverse, TurnRight and TurnLeft. BBC software components require 12 input ports to receive data from six behaviours: Avoid, Stop, Forward, Reverse, TurnRight and TurnLeft software components behaviour. Two output ports from the BBC software component were sent to the motorctrl_left software component while two other output ports were sent to motorctrl_right software component and one output port was sent to the HCI component.

5.4 Composition and implementation phase

Fig. 20 illustrates the I-Wheelchair component composition which includes 11 active and seven passive components, consisting of eight leaf components (without sub-component) and nine composite components (with sub-component) as shown by the blocks with shadow. Every single component provides the ports and connection lines to show the overall composition of the intelligent wheelchair.

From this composition diagram, code template was generated based a Component-Oriented Programming (COP) framework as proposed in Jawawi et. al (2007). The code consists of three block codes, which are the synchronization part, execution part, initialization part. The synchronization part synchronizes the connection port, the execution part chooses a behaviour based on the current command and the initialization part gives a default value.

The COP is supported by an experimental component composition tool. Fig. 21 shows an interface of component composition using the COP tool and all the new and reusable components from other application are listed on the right hand site of the composition. The COP tool is still at development stage, current version of our tool supports component integration, composition and code generation of the structure composition. The designer needs to manually code the functional behaviour of each component. Once the composition is completed, the next step is to analyse the property bundles value such as period and priority for active components using real-time scheduling theory. The scheduling analysis of the composition can be done since the timing required for the analysis were documented with the designed composition.

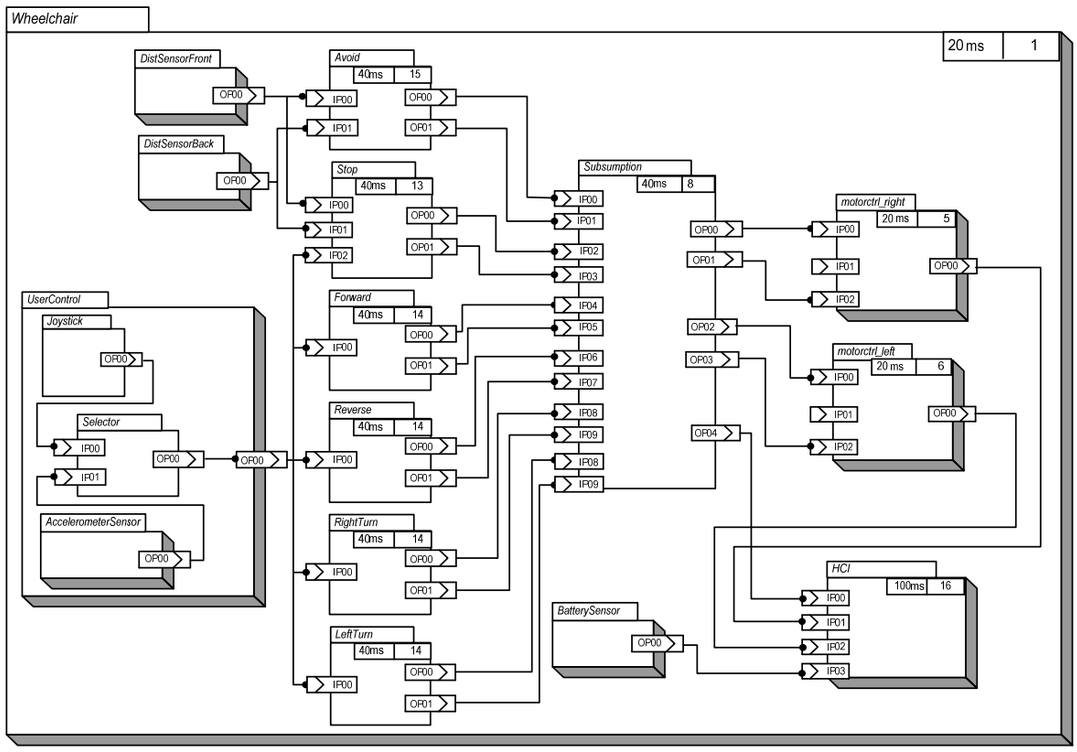


Fig. 20. I-Wheelchair component composition

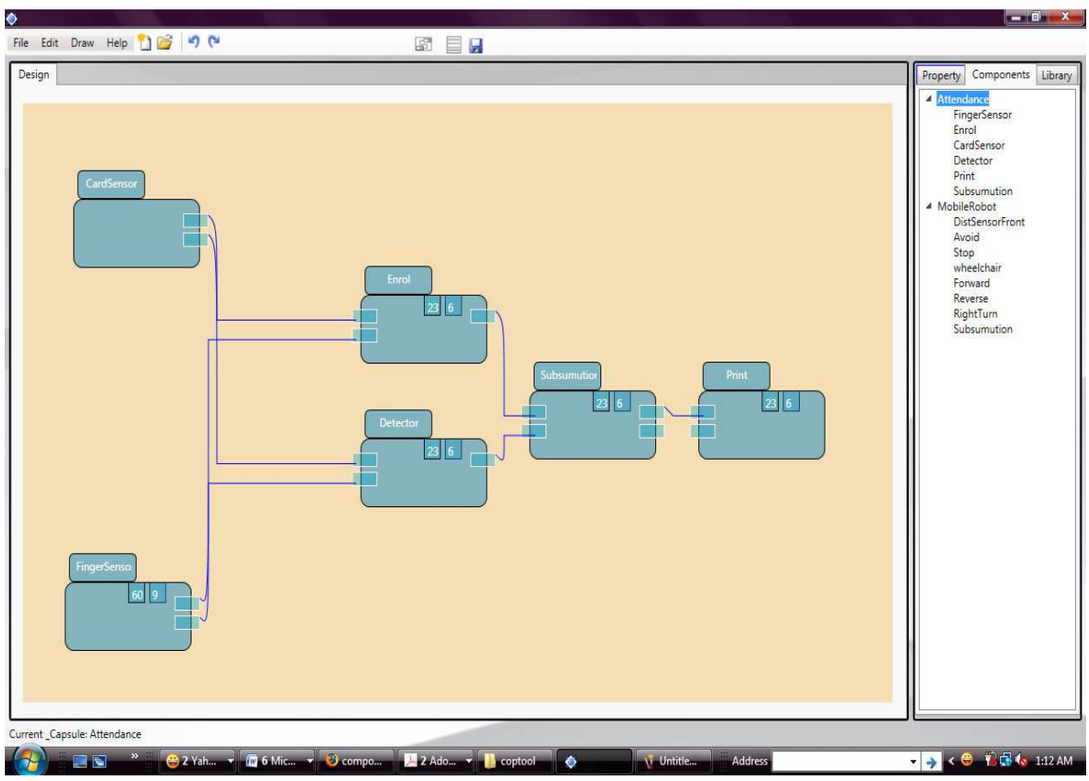


Fig. 21. COP tool's interface to support component composition and code generation

5.5 The software reuse results

The case study application showed how the process model can be used to support the CBD activities of the wheelchair software. The result of implementing these activities on the I-Wheelchair was software with 3583 lines of C codes which generates firmware with code size of about 8.6 Kb with RAM usage of about 1.6 Kb. The software was implemented on an ATMEGA32 8-bit microcontroller with 2 Kb RAM, which illustrated an implementation on a self-contained I-Wheelchair system. This size proves a light-weight solutions integrated in the developed process model is suitable for resource-constrained ERT systems. The real-time performance of the system was predicted during components composition. This prediction was verified in performance testing during implementation phase where the wheelchair showed a reliable behaviour especially on hard real-time tasks.

The design and implementation of the wheelchair software design components were derived from our previous studies and development of mobile robot software systems. One of the studies was the component engineering process of the reusable design components in the Autonomous Mobile Robot (AMR) software development documented in Jawawi et. al (2007). Each component in the reused component repository was modeled using the COP framework.

The reusing process of the software components from an AMR system to the I-Wheelchair aims to analyze the reuse level resulting from the implementation of the process model activities. Amount of reused components were used to assess a reuse improvement effort by tracking percentages of reused components with two types of component reuse: reuse as is and instantiated reuse.

The main concern in this amount of reused components is to measure the reuse of the software components on different platforms and different physical sizes of the AMR and the I-Wheelchair systems. The strategy adopted in analyzing the amount of reused components in this work is to compare the percentage of component reused in two cases:

1. Case 1 Reuse: MobileRobot1 to MobileRobot2 software reuse as reported in Jawawi et. al (2007) and
2. Case 2 Reuse: MobileRobot1 to I-Wheelchair software reuse as shown in this section.

The differences between the three systems are tabulated in Table 2 and Table 3. Table 2 is to differentiate the platform and size of the systems and Table 3 is to differentiate the number of hardware used in the systems. The code used in the table is as: Mtr – Motor, Fan, Enc – Encoder, DS - Distance sensor, PS - Proximity sensor, TS - Temperature sensor, LS - Light sensor, AS - Accelerometer sensor and KP – Keypad.

Case-studies	Processor Type	Size (cm)	Shape	EPROM (Kilobytes)	RAM (Kilobyte)
MobileRobot1	AMD188ES (16 bits)	40	round	64	128
MobileRobot2	ATMEL AVR MEGA32 (8 bits)	16	octagonal	32	2
I-Wheelchair	ATMEL AVR MEGA32 (8 bits)	Standard size	Standard manual	32	2

Table 2. The AMR and wheelchair systems processor platform and size

From Table 3, two hardware components exist in both systems are motor and distance sensor. The types of distance sensor used in three case-studies are the same but the types of motor used in the three cases studies are different.

Case-studies	Mtr	Fan	Enc	DS	PS	TS	LS	AS	KP
MobileRobot1	2	0	2	2	2	0	0	0	0
MobileRobot2	2	2	0	1	0	1	1	0	0
I-Wheelchair	2	0	0	6	0	0	0	1	1

Table 3. The AMR and wheelchair systems sensors and actuators

To measure the software reuse rate, the number of reused component from the components repository will be analyzed. The parameters identified to analyze the amount of reused components in the two reused cases are: reused 100% from design components repository without changes (reuse as is), new components developed (new) and reused components by instantiating the components from MobileRobot1 for the MobileRobot2 or the I-Wheelchair software development (instantiated reuse). Table 4 shows the number of reused components in the two reused cases and the components are grouped into five groups according to our components repository groups.

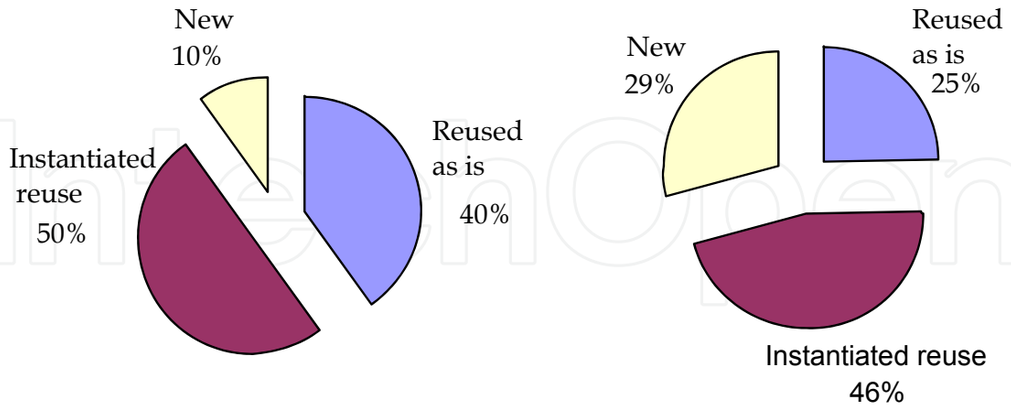
Component Groups	MobileRobot1 to MobileRobot2			MobileRobot1 to I-Wheelchair		
	Reused as is	**Instantiated reuse**	**New**	**Reused as is**	**Instantiated reuse**	**New**
Sensors	2	1	1	2	1	1
Actuators and Motor Control	0	5	0	0	4	0
Behavior and Subsumption	3	1	1	2	2	3
Human-Robot Interfaces	3	0	0	2	0	3
Input-Output	0	3	0	0	4	0
Total components	**8**	**10**	**2**	**6**	**11**	**7**

Table 4. The number of reused and new components in the reuse cases

Table 4 shows higher number of components reused without changes in robot to robot reused case as compared to robot to wheelchair reuse case. This was due to the differences in the components in HRI group and Behavior and Subsumption group. The wheelchair system required different HRI devices and the behavior of wheelchair is not fully autonomous, which still require human control to navigate the wheelchair system. It led to more number of new behavior components or changes to the existing behavior components. Apart from HRI group and Behavior and Subsumption group, other groups showed the same reused pattern.

The analysis work was to identify the rate of components reused from the design repository integrated in the proposed process model. The component reused percentage is the calculation of percentage of reused software components over total software components in the composition. The summary of the component reused percentage for both reused cases are shown in Fig. 22. The figure shows up to 90% component reused rate achieved in designing MobileRobot2 software from the MobileRobot1 design components and 71% components reuse achieved in designing I-Wheelchair software from MobileRobot1 components. It showed that the reuse as is percentage for Case 1 is higher than Case 2. The instantiated reuse rate in both reused cases were about the same but the reused as is rate

was higher in Case 1. This was due to high number of new components required in the wheelchair system as shown in Table 4.



(a) Case 1: MobileRobot1 to MobileRobot2 (b) Case 2: MobileRobot1 to I-Wheelchair

Fig. 22. Component reused percentage in the two reused cases

The amount of component reuse analyzed in this implementation aims to study the possibility to achieve software reuse from mobile robot software to robotic wheelchair software. The component reused results showed a high rate of component reused was shown in mobile robot to wheelchair reused case, but this rate is lower than mobile robot to mobile robot reused case. This high rate was possible in this implementation due to the same behavior-based control paradigm (Brooks, 1986) and sensors' types used in both wheelchair and mobile robot systems. The systematic process model used in this wheelchair implementation help the wheelchair developers to identify the possible reused component in at the early stage of the software development.

6. Comparison results of component-based development process models

This section discusses the results comparison between the integrated MARMOT and PECOS method with other method, based on the criteria as shown in Table 5.

No.	Groups	Criteria
1	Evaluate the domain application of CBD method	Application domain
2	Evaluate the representation of component in the respective phase of the development process	Component representation in analysis Component representation in design Component representation in implementation
3	Evaluate which process development is supported in the CBD method	Component support in a development process Reusability
4	Evaluate the supportive tool, modelling techniques used and supportive ERT requirements of the CBD methods	Modelling techniques use Tool support Multidisciplinary support Multiple constrain support

Table 5. Summary of the evaluation criteria

The comparisons were made on two existing CBD methods; COMponent-based design of software for Distributed Embedded Systems – version II (COMDES-II) (Ke et al., 2007) and ACCORD/UML (Gerard et al., 2002). ACCORD/UML's targeted audience includes engineers who are not software experts and COMDES-II is a component based software framework intended for efficient development of distributed embedded control systems with hard real-time requirements. Table 6 shows the evaluation summary of CBD methods based on the evaluation criteria.

Criteria/CBD Methods	**COMDES-II**	**ACCORD**	**Integrated MARMOT & PECOS**
Application Domain	Hard real time requirement	Real time application engineering for engineer (not software expert)	Multi-disciplinary and multi-constraint development
Component representation in analysis	Actor Model	DAM (detailed Analysis Modelling) model	Preliminary containment hierarchy
Component representation in design	Not applicable	Not applicable	Block diagram
Component representation in implementation	Function Block	DAM (detailed Analysis Modelling) model	Graphical component composition and code template
Modelling techniques use	Actor model and function block model	UML with extension	UML 2.0 and COP composition
Tools support	Workbench	UML based tool	Any tool that supports UML 2.0, and COP tool
Component support in a development process	Design and implementation	Analysis and prototyping	Analysis, design and implementation
Reusability	Function block model	State-machine inheritance	Detailed design component block and code template
Multidisciplinary support	Software	Software	Hardware and software
Multiple constraints support	Concurrency, time multitasking	High level concurrency	Timing Analysis

Table 6. Comparison of CBD methods

Based on the general criteria evaluation results, it showed all three methods support component at analysis phase and implementation phase. Only the proposed integrated method represent components at design phase using block diagram introduced by MARMOT to be included as part of it containment hierarchy diagram and represented in component composition diagram at implementation phase.
COMDES-II and ACCORD methods combine the design phase component representation with the analysis phase. The integrated method defines the component representation in each phase, this enable all multi-disciplines experts for a system like the robotic wheelchair systems to define and specify their own components. Both the integrated method and ACCORD modelling techniques use UML except COMDES-II method that using actor model and function block model. The UML modelling enables multi-disciplines engineers to communicate and share their software components.
The integration of MARMOT and PECOS is shown to support both multi-disciplinary and multi-constraint real-time requirement. ERT CBD methods criteria showed that only the integrated MARMOT and PECOS method supported multi-disciplinary including software and hardware and multi-constraint ERT requirement. COMDES-II and ACCORD only support software disciplinary element. All three CBD methods support different level of real-time constraints; for example ACCORD supports high level concurrency, the integrated method supports timing analysis and COMDES-II supports concurrency and time multitasking. This is due to different nature of ERT systems targeted by different methods and different real-time requirements supported such soft real-time, hard real-time, time-triggered or event triggered.
The integrated MARMOT and PECOS aims to produce a systematic CBD process model of ERT system where the development process model can support multi-disciplinary knowledge and multi-constraint extra-functionality requirement. The component representation systematically supported all phases: the analysis, design and implementation phases, this can provide more reusability facilities in all phases.
The systematic CBD process model allows the multi-disciplines experts in the robotics wheelchair CBD development to develop their reusable components and integrates their components with other. Among the software components developed by different discipline reused in the I-Wheelchair system development are such as BBC component, motors component, sensors and input-output components.

7. Conclusion

This chapter proposed a method to enable reuse of mobile robot software into a robotic wheelchair (I-Wheelchair) software system. The robotic wheelchair system prototype was developed to test the implementation result of the proposed method. The component-based development (CBD) of the robotic wheelchair software using a set of reusable components, and the software composition of the wheelchair software were illustrated. The integration of PECOS into MARMOT method would produces a systematic CBD process model in terms of multi-disciplinary knowledge and multi-constraint extra-functionality requirement. The application of the CBD method in the I-Wheelchair system illustrated how the MARMOT and PECOS were integrated to produce a systematic development method for CBD. The method guides developers to model and deploy the robotic wheelchair software using the

CBD activities and the phases are defined by the process model. With the strengths of MARMOT and PECOS integrated into the method, it enabled the functional and non-functional requirements, especially the timing properties, to be modelled explicitly in all phases. Furthermore, the method also considered multi-disciplinary requirements in its models which help software engineers to focus on their problem-solving of the robotic wheelchair system development. The activities to reuse software from existing robotics systems were explicitly supported in the proposed process model.

The implementation of the I-Wheelchair case study, demonstrated that the proposed method has helped to guide the CBD of the software system from the analysis of the product to the implementation phase through the defined process model. The component reused rate achieved in the wheelchair implementation from a mobile robot design repository was high up to 71% and this implementation was compared with components reuse from the same mobile robot design repository to another mobile robot system to identify the pattern of reuse in both reused cases. No direct relation between the process model proposed and the reused rate studied in this chapter but the process model helped developer to identify and implement the component in all CBD phases. For future works, the existing component-based development tools will be refined further to support COD and COP methods and to integrate the tools with the UML models produced in the COA method.

8. Acknowledgment

Special thanks to the Universiti Teknologi Malaysia for financing and funding this research through Research University Grant and also to our Embedded & Real-Time Software Engineering Laboratory (EReTSEL) members for their continuous support.

9. References

Atkinson, C.; Bayer, J.; Bunse, C.; Kamsties, E.; Laitenberger, O.; Laqua, R.; Muthig, D.; Paech, B.; Wust, J. & Zettel J. (2002). *Component-Based Product Line Engineering with UML*, Addison Wesley Professional, ISBN 978-0201737912, Boston, USA.

Bonail, B.; Abascal, J. & Gardeazaba,l L. (2009). Wheelchair-based Open Robotic Platform and its Performance within the AmbienNet Project, *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '09)*, ISBN 978-1-60558-409-6, Corfu, Greece, June 09-13, 2009.

Brooks R.A. (1986). A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation. RA-2(1): 14-23.

Bunse, C. & Gross, H. G. (2006), Unifying Hardware and Software Components for Embedded System Development, In: *Architecting Systems with Trustworthy Components*, Reussner R. H., Stafford J. A. and Szyperski C. A., pp. 120-136, Springer-Verlag, ISBN 978-3540358008, Berlin, Germany.

Carlsona, J.; Håkanssonb, J. & Pettersson, P. (2006). Save CCM: An Analysable Component Model for Real-Time Systems, *Proceedings of the International Workshop on Formal Aspects of Component Software (FACS 2005)*, vol. 160 , Macao, October 24-25, pp.127-140.

Cheein, F. A. A.; Cruz, C.; Bastos, T. F. & Carelli, R. (2009). SLAM-based Cross-a-Door Solution Approach for a Robotic Wheelchair. *International Journal of Advanced Robotic Systems*, Vol. 6, No. 3. pp. 239-248, ISSN 1729-8806.

Dogru, A. H. & Tanik M. M. (2003). A Process Model for Component-Oriented Software Engineering. *IEEE Software,* Vol. 20 No. 2, (March 2003), pp. 34-41.

Gerard, S; Terrier, F. & Tanguy, Y. (2002). Using the Model Paradigm for Real-Time Systems Development ACCORD/UML. *Proceedings of the Workshops on Advances in Object-Oriented Information Systems (OOIS '02)*, Montpellier, France, September 2, pg. 260-269.

Gong, J.; Peng, Y.; Hao, J.; Huang J., & Huang, K. (2010), Research on Component-Oriented Methodology for Constructing Simulation Systems. *Journal of System Simulation*, pp. 1-10.

Jang, C; Lee, S-I; Jung, S-W; Song, B; Kim, R; Kim, S & Lee, C-H. (2010). OPRoS: A New Component-Based Robot Software Platform. *ETRI Journal*, Vol. 32, No. 5, (October 2010), pp 646-656.

Jawawi, D.N.A; Deris, S. & Mamat, R. (2006). "Enhancements of PECOS Embedded Real-Time Component Model for Autonomous Mobile Robot Application". *Proceeding of The 4th ACS/IEEE International Conference on Computer Systems and Applications*, pp. 882 – 889. Dubai/Sharjah, March 8-11 2006.

Jawawi, D. N. A.; Mamat, R. & Deris, S. (2007). A Component-Oriented Programming for Embedded Mobile Robot Software. *International Journal of Advanced Robotic Systems*, Vol. 4, No. 2, (September 2007), pp. 371-380. ISSN 1729-8806.

Ke, X.; Sierszecki, K. & Angelov, C., (2007). COMDES-II: A Component –Based Framework for Generative Development of Distributed Real-Time Control Systems, *Proceeding of 13th IEEE International Conference on Embedded Real Time Computing System and Applications (RTCSA 2007)*, pp. 199 – 208, ISBN 978-0-7695-2975-2, Daegu, Korea, *August 21-23*, 2007.

Lee, S. C. & Shirani A. I. (2004). A Component Based Methodology for Web Application Development. *Journal of Systems and Software*, Vol.71, No. 1-2, (April 2004), pp.177-187.

Mallet A.; Kanehiro F.; Fleury S. & Herrb M. (2007). Reusable Robotics Software Collection. *Proceedings of Second Int. Workshop on Software Development and Integration in Robotics (SDIR)*. Roma, Italy, April 2007.

Nesnas I. A.D.; Simmons R.; Gaines D.; Kunz C.; Diaz-Calderon A.; Estlin T.; Madison R.; Guineau J.; McHenry M.; Shu I-H. & Apfelbaum D. (2006). CLARAty: Challenges and Steps Toward Reusable Robotic Software. *International Journal of Advanced Robotic Systems*, Vol. 3, No. 1, (2006), pp. 23-30. ISSN 1729-8806.

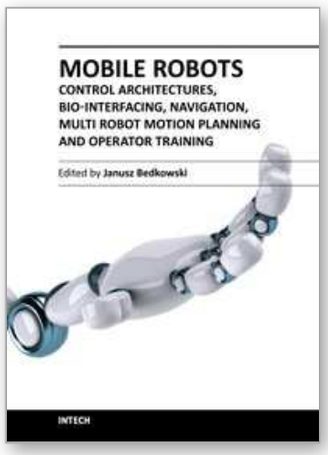
Nierstrasz, O.; Arévalo, G.; Ducasse, S.; Wuyts, R.; Black, A.; Müller, P.; Zeidler, C.; Genssler, T. & van den Born, R. (2002). A Component Model for Field Devices. *Proceedings First International IFIP/ACM Working Conference on Component Deployment*, pg. 200-209, June 20-21, ISBN 3-540-43847-5. Berlin, Germany, June 20-21, 2002.

Ommering, R.; Linden, F.; Kramer, J. & Magee, J. (2000). The Koala Component Model for Consumer Electronics Software. *IEEE Computer*, Vol. 33, No. 3, (Mar 2000). pp. 78 – 85, ISSN 0018-9162.

Schuppenies, R. & Steinhauer, S. (2001). Software Process Engineering Metamodel, OMG group, November 2002.

Simpson, R.; Lopresti, A. E.; Hayashi, S.; Nourbakhsh, I. & Miller, D. (2004). The Smart Wheelchair Component System. *Journal of Rehabilitation Research & Development*. Vol. 41, No. 3B, (May/June 2004), pp 429–442.

Wang, A. J. A. & Qian, K. (2005). *Component-Oriented Programming*. Wiley-Interscience, ISBN 0471644463, Danvers, USA.



Mobile Robots - Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator Training

Edited by Dr. Janusz Będkowski

ISBN 978-953-307-842-7
Hard cover, 390 pages
Publisher InTech
Published online 02, December, 2011
Published in print edition December, 2011

The objective of this book is to cover advances of mobile robotics and related technologies applied for multi robot systems' design and development. Design of control system is a complex issue, requiring the application of information technologies to link the robots into a single network. Human robot interface becomes a demanding task, especially when we try to use sophisticated methods for brain signal processing. Generated electrophysiological signals can be used to command different devices, such as cars, wheelchair or even video games. A number of developments in navigation and path planning, including parallel programming, can be observed. Cooperative path planning, formation control of multi robotic agents, communication and distance measurement between agents are shown. Training of the mobile robot operators is very difficult task also because of several factors related to different task execution. The presented improvement is related to environment model generation based on autonomous mobile robot observations.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen