

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

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

SIMBA ANAK BAU

A project report submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computer Science and Information System
University of Technology Malaysia

OCTOBER 2008

ACKNOWLEDGEMENT

In appearing this project, I wish to express my sincere appreciation to my supervisor Dr. Dayang Norhayati Binti Abang Jawawi for the guidance, advice and encouragement during my studying. The support and suggestion that Dr. Dayang give me the inspiration to going through in this project.

I would like to thanks the Software Engineering Lab members in Universiti Teknologi Malaysia for their helps and support.

Finally, my special thanks to my parent for their love and care specially my mother, my sisters and my brothers, and also my special friend Tony for your support and cheering me up at those difficult time.

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 PErvasive 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 kompleks, 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 kompleks. Berdasarksn perspektif tersebut, projek ini bermatlamat untuk membolehkan pembangunan sistem masa nyata berasaskan corak yang dipanggil Pattern-Oriented Analisis 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		PAGE
	DECLARATION OF STATUS THESIS	
	SUPERVISOR DECLARATION	
	TITLE PAGE	i
	STUDENT DECLARATION	ii
	ACKNOWLEDGEMENT	iii
	ABSTRACT	iv
	TABLE OF CONTENT	vi
	LIST OF TABLE	x
	LIST OF FIGURE	xi
	LIST OF ABBREVIATION	xiii
	LIST OF APPENDIX	xiv
1	PROJECT OVERVIEW	
	1.1 Introduction	1
	1.2 Problem Background	4
	1.3 Problem Statement	6
	1.4 Project Aim	6
	1.5 Objectives	7
	1.6 Scopes	7
	1.7 Significance of the project	8

2	LITERATURE REVIEW	
2.1	Introduction	9
2.2	Pattern-oriented methodology	10
	2.2.1 Pattern-Oriented Analysis & Design (POAD)	10
	2.2.2 Pattern-Driven Modeling & Analysis (PDMA)	14
	2.2.3 Metamodel POAD and PECOS	15
	2.2.4 Component-oriented pattern	16
	2.2.5 Design pattern and CBSD	17
	2.2.6 Summary of pattern-oriented methodology	19
2.3	Component-Oriented Technology	21
	2.3.1 PECOS	21
	2.3.2 COM	24
	2.3.3 CORBA	25
	2.3.4 .NET	27
	2.3.5 Summary of component-oriented technology	29
2.4	Graphical Programming	30
	2.4.1 LabVIEW	31
	2.4.2 UML-RT	32
	2.4.3 Simulink	34
	2.4.4 Summary of graphical programming	35
2.5	Software Process	36
	2.5.2 Software Process Engineering Metamodel	37
2.6	Summary	39
3	RESEARCH METHODOLOGY	
3.1	Introduction	40
3.2	Operational Framework	40
	3.2.1 Analysis problems and conduct literature review	42
	3.2.2 Propose project	42
	3.2.3 Project planning	43
	3.2.4 Identify and study POAD and PECOS	43

3.2	Hardware and software requirement	44
3.5	Project schedule	45
3.6	Autonomous Mobile Robot Case Study	45
3.7	Summary	47
4	POAD AND PECOS PROCESS MODEL	
4.1	Introduction	48
4.2	The Software Process Engineering Meta Model	48
4.3	The Process model	50
4.3.1	Use Case Diagram	51
4.3.2	Analysis Phase	51
4.3.3	Early Design Phase	54
4.3.4	Detailed Design Phase	59
4.4	Discussion on Process Model	62
5	PROCESS MODEL USING UML-RT	
5.1	Introduction	63
5.2	Mapping Process	63
5.2.1	Mapping POAD into UML-RT	64
5.2.2	Mapping UML-RT into PECOS Model	67
5.3	Process Model	69
5.3.1	Analysis Phase for AMR	69
5.3.2	Early Design Phase for AMR	73
5.3.3	Detailed Design Phase for AMR	81
5.4	Discussion on Process Model using UML-RT	82
6	CONCLUSION	
6.1	Summary	84
6.2	Project Architecture and Contribute	85
6.3	Future Work	86

REFERENCES

87

APPENDIX

91

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
2.1	Comparison of Pattern-Oriented Methodology	20
2.2	Comparison of Component-oriented technology	30
2.3	The comparison of Graphical programming	36
2.4	Stereotypes of SPEM	38
5.1	Mapping Component-Based Analysis Pattern and Use Case	71

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	The POAD process phase	11
2.2	The POAD analysis phase	12
2.3	The POAD design process	13
2.4	Overview software process for POAD and PECOS	16
2.5	CBSD using Design Pattern	18
2.6	PECOS Component Model [Nierstrasz,2002]	22
2.7	Instances creation and dynamic loading of code in COM	25
2.8	CORBA Component [Wang & Qian,2005]	26
2.9	NET Framework [Wang & Qian,2005]	28
2.10	An example of UML-RT application	33
2.11	Simulink Original Model [Zhan & Clark,2005]	35
3.1	Operational Framework	41
3.2	Block Diagram of the APIBOT	46
4.1	SPEM Icons	49
4.2	The disciplines of POAD and PECOS	51
4.3	Phases in POAD +PECOS process	51
4.4	Use case diagram for analysis phase	52
4.5	Use case diagram for early design phase	53
4.6	Use case diagram for detailed design phase	53
4.7	POAD + PECOS Analysis phase	55
4.8	The Analysis phase describe in terms of works definition and	56

	work product	
4.9	POAD and PECOS Early design phase	57
4.10	The early design phase described in terms of works definition and work product	58
4.11	The detail design phase described in terms of works definition and work product	60
4.12	The activity of detail design phase	61
5.1	Mapping Pattern Level diagram into UML-RT	65
5.2	Mapping Pattern Level diagram with Interface into UML-RT	66
5.3	Mapping Pattern Level diagram with Control Interface into UML-RT	67
5.4	Mapping UML-RT and PECOS	68
5.5	The AMR use case diagram	70
5.6	Pattern level diagram for AMR	73
5.7	Pattern level diagram with interface for AMR	74
5.8	Example interface for Distance Sensor	74
5.9	Pattern level diagram with control interface for AMR	75
5.10	Detail Control Interface for Pattern Level diagram	76
5.11	The Behavior of AMR design using UML-RT	77
5.12	Sensor Protocol between Avoid and Distance	78
5.13	Structure Diagram for Avoid	78
5.14	Structure diagram for Distance	79
5.15	Input Port after receive input	79
5.16	State diagram for avoid before receive input	80
5.17	State diagram avoid with input NoObstacle	80
5.18	State diagram avoid with input ExistObstacle	81
5.19	The APIBOT composition based in design component artifact	82

LIST OF ABBREVIATIONS

CBSE	Component-Based Software Engineering
CBSD	Component-Based Software Development
COT	Component-Oriented Technology
POAD	Pattern-Oriented Analysis and Design
ERT	Embedded Real-Time
PECOS	PErsive COmponent System
COM	Component-Oriented Model
PDMA	Pattern-Driven Modeling and Analysis
CORBA	Common Object Request Broker Architecture
COP	Component-Oriented Programming
UML-RT	Unified Modeling Language- Real-Time
SOAP	Simple Object Access Protocol
IDL	Interface Definition language
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
FP	Front Final
BD	Block Diagram
SPEM	Software Process Engineering Metamodel
CMMI	Capability Maturity Model Integration

LIST OF APPENDIX

APPENDIX	TITLE	PAGE
A	Project 1	91
B	Project 2	92

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 made executable, which can be assembled with other component, which can be reused as a unit in various contexts (Wang and Qian, 2005). PErvasive 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 web-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. PErvasive COmponent Systems (PECOS) as a component technology.
- iii. The implementation only for medium size of Embedded Real-Time systems.

1.7 Significance of the project

The significant of this project is to enable or to promote pattern reused in ERT systems. The study is focus on software process for integrated POAD and PECOS Meta model. Based on the understanding of the integration process of POAD and PECOS Meta model, the process will be identified and defined. The additional knowledge can be achieved through the analysis process, where before identified and defining the process of the integration, the researcher needs to study about the POAD and PECOS. There are many advantages can be carried out from this study, other than promoting pattern reused in ERT system, the study also can improve the knowledge about pattern oriented and component oriented.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In order to identify and define the software process for integrated of POAD and PECOS on ERT software development, some literatures concerned on pattern-oriented methodology and component-oriented technology need to be covered. The graphical modeling needs to be reviewed, which concerns about graphical tools that are normally used to model the application for ERT software development only. The main objective of this project is to identify and define the process for integrated POAD and PECOS into formal form. Before going on more detail in this project, the software process is one of element are need to study.

2.2 Pattern-Oriented Methodology

Pattern is introduced to document good design practices, which a vehicle of knowledge and experience transfer from expert to the novice. There are several kinds of software patterns, including analysis pattern, architecture pattern and design pattern, but the most popular is design pattern (Yacoub and Ammar, 2004). Pattern methodology is an approach that is used to develop software application in analysis and design stages. The pattern-oriented methodology includes:

2.2.1 Pattern-Oriented Analysis & Design (POAD)

Pattern-Oriented Analysis & Design (POAD) is a methodology which is usable at analysis and design stage. POAD is a software development approach based on structural composition of design pattern. The main objective of POAD is to provide a solution to improve the practice systematically deploying design pattern in application development. The documentation of design pattern describes detail about the pattern, its usage, structure, behavior of participants, forces and consequences and guidelines for implementation (Yacoub and Ammar, 2004). The POAD methodology is introduced to overcome these problems with a structural approach to glue patterns at the high level design and use the notion of constructional design patterns as design patterns with interfaces.

In general, the POAD methodology has three phases of process, which are analysis phase, high-level design phase and design refinement phase (Yacoub and

Ammar, 2004). POAD process explains the processes and steps to develop an application design using pattern. The Figure 2.1 illustrates the phase of POAD.

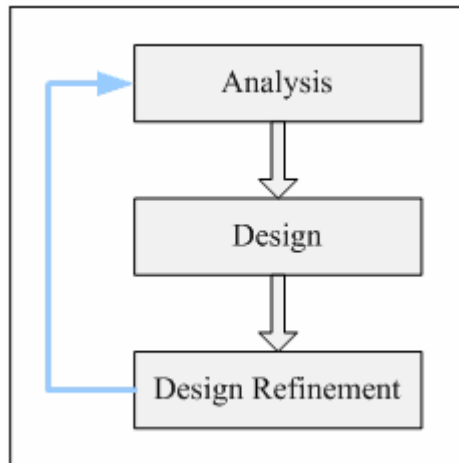


Figure 2.1: The POAD Process phase

The purpose of analysis phase is to analyze the application requirement and decide on set of design pattern that will be used in designing system. The processes at analysis phase are analyzing the application requirement and model it into UML use case diagram and sequence diagram. The product from this phase is set of design pattern chosen by the application analyst. The Figure 2.2 illustrates the process in analysis phase.

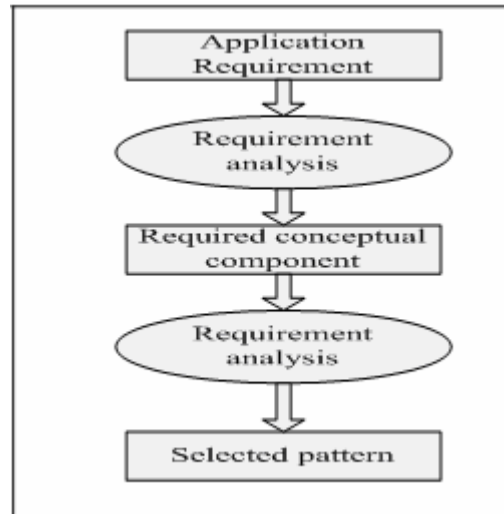


Figure 2.2: The POAD analysis process

The second phase in POAD called design phase, this phase aim to develop the application design by composing the pattern that has been selected in the analysis phase. The activity involve in design phase is create the instance of the pattern that were selected in the analysis phase and identifies the relationship between the instance. The products that produce by design phase called Detailed Pattern Level diagram and Figure 2.3 illustrates the process which involve in design phase. The last phase in POAD is Design refinement phase, the purpose of this phase is to develop the profound and class diagram for the application. The process start with the detailed pattern-level diagram, the designer choosing names for pattern participants that are meaningful in the applications context and defining application-specific names for operations in the pattern class. The product that produce called optimized class diagram for the application.

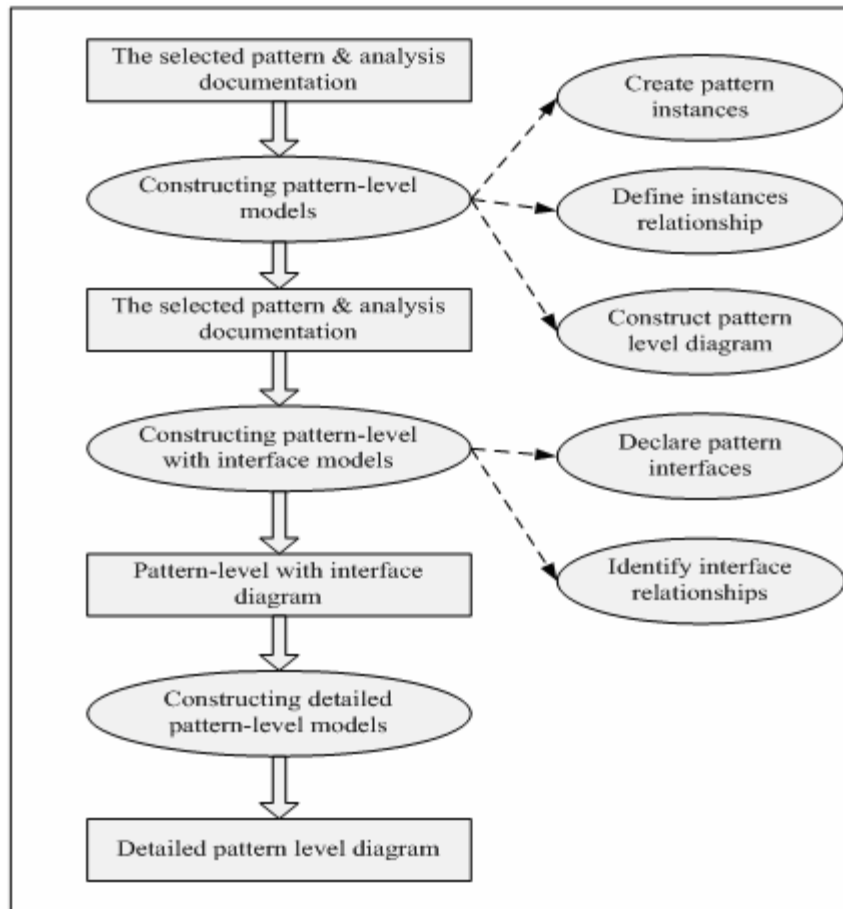


Figure 2.3: The POAD design process

The advantage with applying the POAD development process in object-oriented design is that the refractory activities can be reduced because the part of refactoring process has already been taken care of in the pattern themselves. Using POAD the documentation of the application or framework is better because pattern-level diagram provides a high-level abstract view of the design and POAD provides a solution to traceability problems. The disadvantage when applying POAD is the designer needs to understand the tradeoff, motivations, forces and related pattern to make a correct decision when choosing a pattern in the pattern-oriented design.

2.2.2 Pattern-Driven Modeling and Analysis

Pattern-Driven Modeling and Analysis (PDMA) are another methodology which based on pattern. The PDMA approach (Konrad, 2004) is a pattern-driven for modeling and analyzing requirements of embedded systems based on the concept of object analysis pattern. The object analysis pattern in PDMA approach uses UML to model the structural and behavioral information, which can be used to guide the construction of UML-based conceptual model on embedded systems. The objectives of the object analysis patterns is to provide guidance to a new developers on embedded systems to determine the key elements of many embedded systems and provide examples on how to model these elements with UML commonly accepted diagramming notation.

The PDMA approach is supported by MINERVA, Hydra and Spin. The MINERVA is an automated tool for graphical construction of syntactically correct diagram and visualization within UML diagram, which consistency check the results, simulation trace and paths of execution that lead to errors. The Hydra is an automated tool to map UML to a target language, while Spin is a simulation tool to check temporal constraints against the conceptual model. Based on the PDMA approach the developers can accelerate the initial development of conceptual models through using the object analysis pattern and then formalization the works and tools, which means to carefully check whether requirements are capture appropriately in the models using simulation and model checking techniques.

2.2.3 Metamodel POAD and PECOS

The combination Meta model of POAD and PECOS was introduced by Jawawi (2005). The goal of combination between Meta model present by POAD and PECOS is to enable high-reused software in ERT systems. In order to map between these two Meta model Jawawi (2005), was introduced the enhancement on POAD and PECOS. The enhancement is needed because of the different definition of these two approaches, which does not allow direct mapping. The mapping process is concerned on converting the diagram in analysis and early design of POAD into design diagram at PECOS.

Moreover, there are some issues need to be clarified in the transformation process: 1) how to migrate pattern to component, 2) how interfaces in two meta model can be mapped and 3) how real-time behavior can be explicitly expressed in the two models. This issue has been solved by Jawawi (2005). In the process to transform these two Meta models Jawawi (2005) divided the ERT development into three phases. The following figure shows the phases of transforming Meta model.

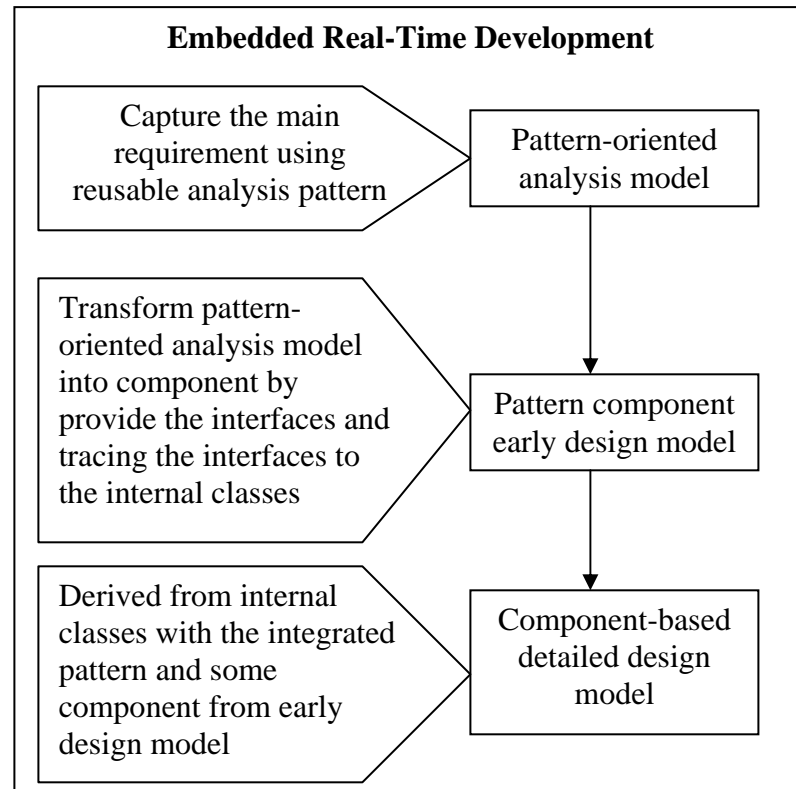


Figure 2.4: Overview software process for integrated POAD and PECOS

2.2.4 Component-Oriented Pattern

The Component-Oriented Pattern is a methodology that has been proposed by Hutchinson and Kotonya (2005). The methodology is an integration of component and pattern, which the concept of pattern was applied into CBSE. The main purpose of this integration is to promote reuse at an even greater level. A pattern and component was appearing as approaches which available to be combine because they have the similarities of behavior. The similarities between the notions of pattern and component: 1) pattern and component both attempt to present reusable solutions to recurring problems, 2) pattern and components can be employed whenever and whatever there is an appropriate match of context and 3) pattern and components can be combined with others pattern or components.

In order to apply pattern into CBSE, there aspect of pattern that good candidate transfer into CBSE was identify: 1) the reuse of experience, 2) the development standard types of solutions, 3) a normalized mechanism for abstracting from multiple examples in order to extract best practices and 4) a means of communication and a method for categorizing problems. The integrated pattern and component is concern with four aspects: 1) high-level architectural pattern, 2) lower-level architectural fragments, 3) concrete component assemblies and 4) service pattern. The high-level architectural pattern defines the pattern in terms of component, connectors, interface ports and the connection between them. The lower-level architectural fragments are used to carrying out high-level design and the interactions between the different sub-systems. The concrete component assemblies are most like instantiated version of the abstract pattern and appropriate candidate for supplying the functionality required for sub-systems in high-level pattern, while the service pattern defined of the pattern-oriented concepts for CBSE development.

2.2.5 Design Pattern and CBSD

The combination of design pattern and CBSD is an approach proposed by Yau and Dong (2000). The goal of this work is to improve software productivity and quality by reusing existing component. The purpose of this approach is to overcome the problems in integrated various components in software system because it is not easy to modify components and some components are black box. This approach is presented to use design pattern to automatic generation of the component wrappers for component integration. Therefore, the services of CBSD can be used by automating component-based software design and implementation.

Moreover, these approaches gives precise representation of abstract solution contained in design pattern and then converts and customize the abstract solutions to concrete solutions in the software design. The concrete solutions are the relationship and interactions among components, which are used to automatic component wrappers generation to integrate component. Figure below shows the CBSD process using design pattern.

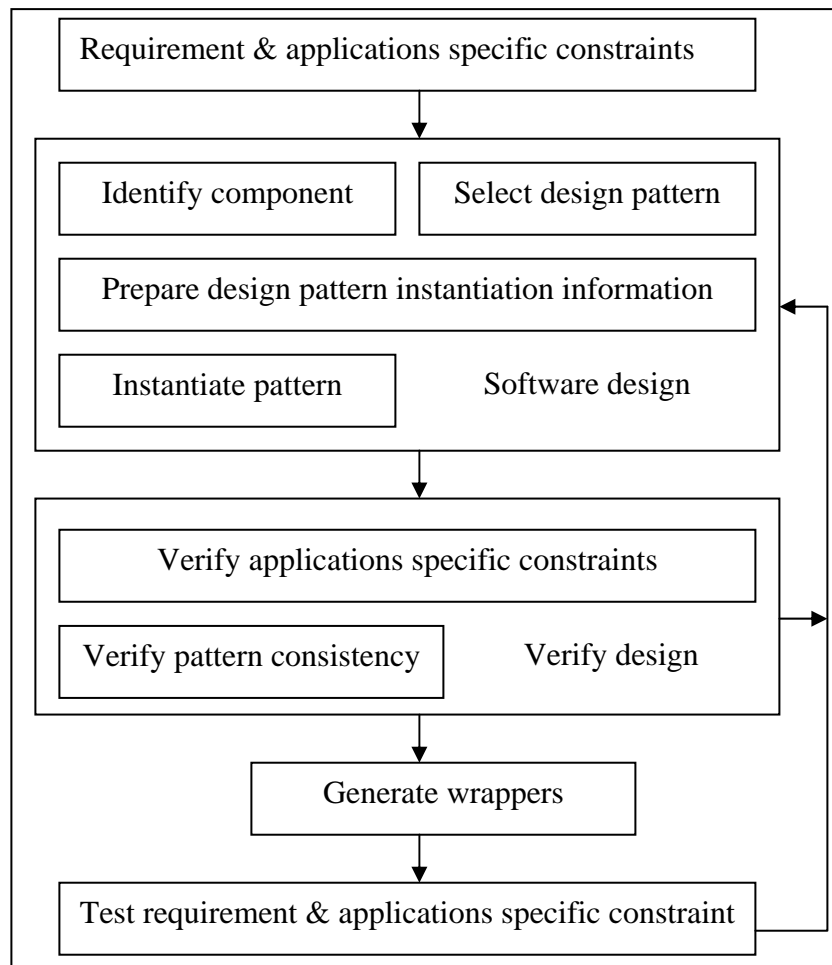


Figure 2.5: CBSD using Design Pattern

2.2.6 Summary of Pattern-Oriented Methodology

Table 2.1 shows the pattern-oriented methodologies that have been reviewed. There are POAD, PDMA, Meta model POAD and PECOS, Component-oriented pattern and Design pattern + CBSD. The methodologies was compare from the six aspects, there are component based, pattern oriented, pattern level, phase level and tools.

Component based and pattern oriented are selected to compare pattern oriented methodology because there two aspects is interested area in this project. A pattern level is selected to compare there methodology because based on pattern level, the types of pattern which is cover are present. Moreover, phase level and tools aspects are selected because based on phase level the completed methodology is show and tools aspect are used to justify any support tools for there methodology.

Based on summary which show at table 2.1, all there methodologies are component based and pattern oriented except for PDMA, which is not component based. The comparison from the phase level element, POAD and Meta model POAD and PECOS are cover the analysis and early design phase, PDMA and component oriented pattern are concern on analysis and design phase, while design pattern and CBSD are focus on design and implementation phase. From the pattern level comparison, POAD, component-oriented pattern and design pattern and CBSD are using design pattern while PDMA and Meta model POAD and PECOS using analysis pattern and all these methodologies do not have their specific tools.

Table 2.1: Comparison of Pattern-Oriented Methodology

Methodology	POAD	PDMA	Metamodel POAD + PECOS	Component- oriented pattern	Design Pattern + CBSD
Component based	Yes	No	Yes	Yes	Yes
Pattern oriented	Yes	Yes	Yes	Yes	Yes
Pattern level	design pattern	Analysis pattern	Analysis pattern	Design pattern	Design pattern
Phase level	Analysis and design phase	Analysis and design	Analysis and design	Analysis and design phase	Design and implementation phase
Tools	No	Yes	No	No	Yes

Based on the review in these pattern-oriented methodologies, the conclusion that can be describe are most of methodology is based on design pattern at analysis and design phase. These shown most of component-based with design pattern has been successful in software engineering, but the approach based on analysis pattern with component has not yet received much attention in software engineering. Based on table at above the analysis pattern is use at Meta model of POAD and PECOS. The benefit when use analysis pattern is to enable capture the conceptual models in those multi-disciplines and analysis pattern consists of component and each component acts as a unit of analysis, which these component will facilities the transformation of the conceptual model into design model, (Jawawi, 2006). Based on there benefit, the analysis pattern is selected to be use in this project.

2.3 Component-Oriented Technology

Component-Oriented Technology (COT) defined as new tools which consist with these behaviors; self-contained, self-deployable computer code with well-defined functionality and can be assembled with other components through its interface (Wang and Qian, 2005). In COT there contain of Component Oriented Programming (COP). COP is enables programs to be constructed from pre-built software components, which are reusable, self contained blocks of computer code, (Wang and Qian, 2005). The infrastructure of COP consists of three models: a component model, a connection model, and a deployment model. A component model defines what a valid component is and how to create a new component under the component infrastructure. COP is a new programming paradigm beyond object oriented programming, which offers higher reusability and better modular structure with greater flexibility (Wang and Qian, 2005). COP is an important technology in software industry because it provides a higher level of abstraction and increases large number of reusable of components libraries. The component-oriented technology includes:

2.3.1 PErvasive COmponent Systems (PECOS)

PErvasive COmponent Systems (PECOS) is the component model, which was introduced to enable component based software development for embedded systems, specifically for field devices (Genssler et al., 2002). A component model in PECOS is for embedded system components addressing behavior specification and non-functional properties and constraints, while a composition language is used to specify components and their compositions, and also supports the interfaces to a composition

environment. A composition environment is used to composing embedded applications from components, validating functional such as interfaces and non-functional compositional constraints such as power consumption and code size. (Nierstrasz et al., 2002)

In order to validate component based software development, PECOS project is developing the hardware and software for a demonstration device. The following Figure shows the model component of PECOS.

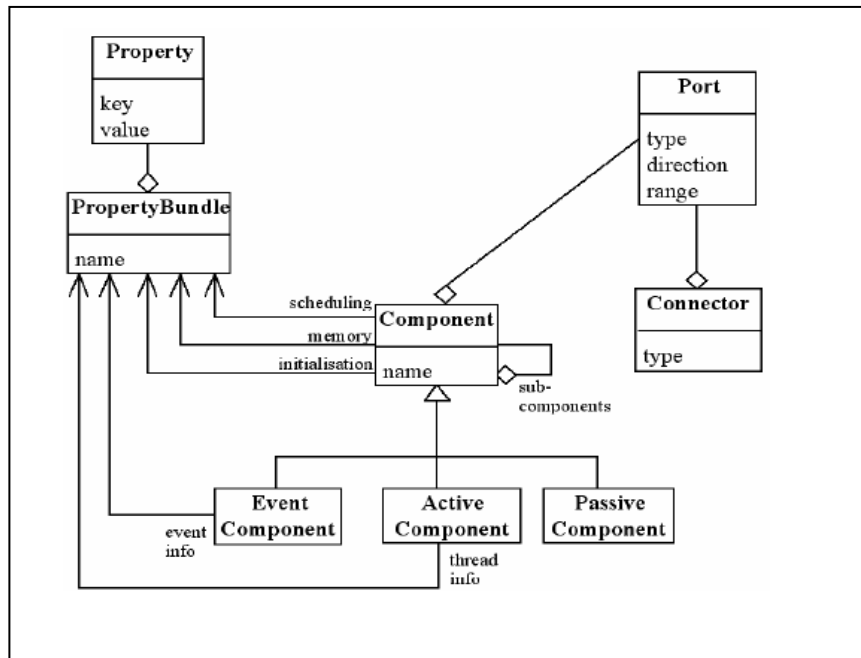


Figure 2.6: PECOS Component Model [Nierstrasz, 2002]

The PECOS field device component model defines the elements of components (Nierstrasz, 2002), there are:

Ports: A ports is a share variable that allows a component to communicate with other components. A port consists of:

Name: has to be unique within the component;

Type: characterizing of data.

Range: defined a minimum and maximum value which can be passed on this port.

Direction: (“in”, “out” or “inout”) indicating whether the component reads (“in”), writes (“out”), or reads and writes (“inout”) the data.

Connectors: A connector specifies a data-sharing relationship between ports, it has a name, a type, and a list of ports it connects.

Properties: A property is a tagged value. The tag is an identifier, and the value is typed.

Property bundle: is a named group of properties, which used to characterize aspects of component such as timing or memory usage.

3 different components: Active Component, Passive Component and Event Component

Active Component: The active components have their own thread of control where used model ongoing or longer-lived activities that do not complete in a short time-cycle.

Passive Component: do not have they own thread of control and the event components. A passive component is explicitly scheduled by the active component that is its nearest ancestor in the component hierarchy. Passive components are typically used to encapsulate a piece of behaviors that executes synchronously and completes in a short time-cycle.

Event Component: are those whose behavior is triggered by an event. They are used to model pieces of hardware that frequently emit events, such as motors that give timers that release a timing event when a certain deadline has passed, whenever the event fires, the behaviors is executed immediately.

PECOS has a two main parts, static structure model and execution model. A static structure model describes the entities included in the model, the features and the

properties, while the execution model defines the behavior of the component execution. The static structure model is a component that can be hierarchically built from other subcomponent, a component can be composite component or leaf component. Composite component contains a number of connected subcomponent and leaf component is a block-box not further defined by model but instead directly implemented in the host programming language. It has an interface consisting of a set of ports, and properties specified by its property bundles. The execution model contains execution behavior and synchronization behavior. The execution behavior describes the action that is performed when component is executed, while synchronization behavior is responsible to synchronize the data flow between components (active component and event component).

2.3.2 Component Object Model (COM)

Component Object Model (COM) is component technologies which allow network-based components interaction. COM provides a framework for creating and using components on a Windows platform, which supports interoperability and reusability of distributed objects by allowing developers to build systems through assembling reusable components from different vendor that, communicate via COM.

COM defines an application programming interface (API) to allow for the creation of components for use in integrating custom applications or to allow diverse components to interact, as long as the components adhere to the binary structure specified by Microsoft. COM defines a dialect of the Interface Definition Language (IDL) that is used to specify object-oriented interfaces. Interfaces are object-oriented in the sense that their operations are to be implemented by a class and passed a

reference to a particular instance of that class when invoked. The mechanism for creating instances of these classes is closely linked with how and when the code in different components is linked together. Figure below shows how to create instance and dynamic loading in COM. (Luders et al., 2007).

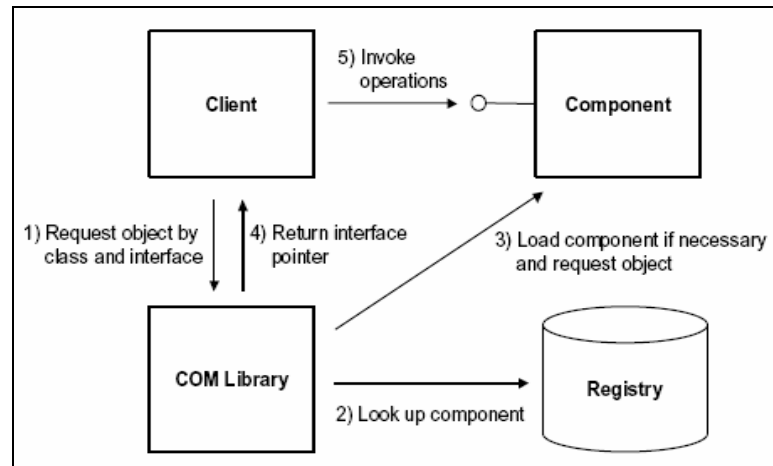


Figure 2.7: Instance creation and dynamic loading of code in COM

2.3.3 Common Object Request Broker Architecture (CORBA)

Common Object Request Broker Architecture (CORBA) is a component technology that is a specification of a standard architecture, which allows the vendor to develop ORB (Object Request Broker) product that supports application portability and interoperability across different programming language, hardware platforms, operating systems and ORB implementations, (Wang and Qian, 2005). A large and growing number of implementation of CORBA is available in the market place, including implementations from major computer manufactures and independent software vendor.

CORBA component allows distributed components to interoperate in a language-independent and platform-independent environment. The CORBA components are self-descriptive, they can be deployed on any compliant servers, portable and scalable. The infrastructure of CORBA defines the interoperations of distributed component objects, which handles a request for services provided by a distributed component.

The CORBA component provides service through its interface which describe by an IDL program. IDL is a definition language with its own syntax and semantics and supports a number of data types such as long, short, float, double, char, and boolean. The IDL interface of a CORBA component description all operations it provides and all attributes that clients can use *get* and *set* methods to access them. The IDL interface is implemented by classes of CORBA component and is aware to any client of the CORBA component. It is a contract between a client and the server component. The following figure shows the basic structure of a CORBA component object.

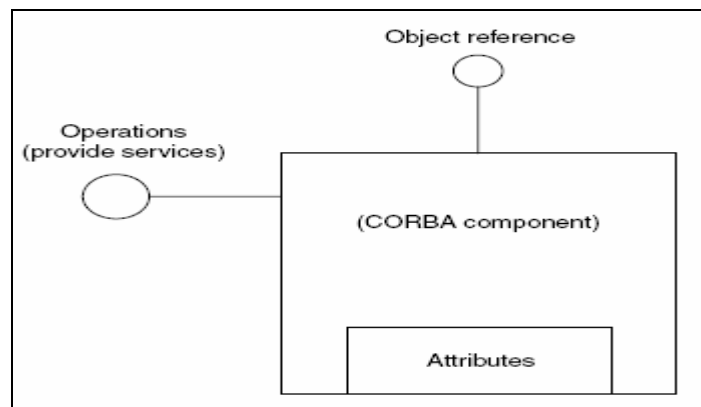


Figure 2.8: COBRA Component [Wang & Qian, 2005]

2.3.4 .NET

The .NET is one of the newest technologies introduced by Microsoft Corporation, which is a platform for rapid and easier building, deploying, and running secured. The .NET software components can be integrated in applications for rapid developing XML Web Services and applications. It provides a highly productive, component-based, multilanguage environment for integrating existing applications with the Internet to meet the challenges of new applications for deployment and operation of internet-scale applications. (Wang and Qian, 2005)

The .NET framework encompasses a virtual machine that provides a new platform for software development. The core of the .NET framework includes XML and Simple Object Access Protocol (SOAP) to provide Web Services over the Internet. The purpose of the .NET framework is to facilitate the developments of desktop window, and Web-based application services on Windows platform and make them available and accessible not only on Windows platform but also on other platforms through common protocols such as SOAP and HTTP. The following figure shows the .NET framework.

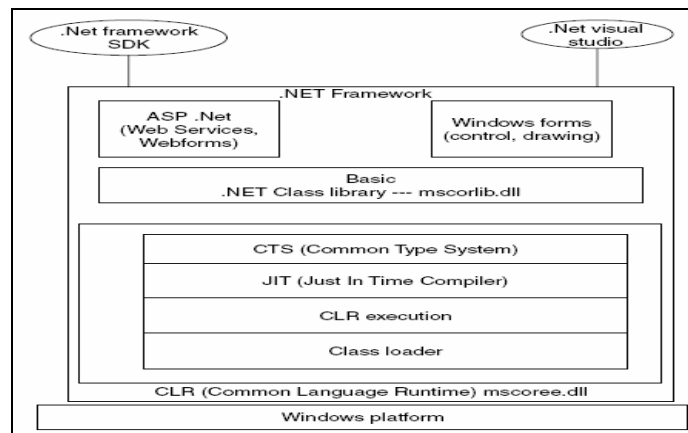


Figure 2.9: .NET Framework [Wang & Qian, 2005]

A .NET component technology is unified language oriented and any .NET component is in the format of precompiled MSIL, which can be binary plugged-in by any other MSIL components or any other .NET compatible clients. A .NET component is a single precompiled and self-described MSIL module built from one or more classes or multiple modules deployed in a DLL assembly file. An assembly consists of up to four parts, there are Manifest name of assembly, Metadata of modules, IL code of modules and Resources such as image files.

A .NET component can be installed at client site, server site, or middleware site. It does not matter what kind of component it is a .NET component always provides some services to its clients, which the client may be another component or client application. A .NET component can be a local component, which can only be accessed locally within the same application domain, in the same machine or a remote component, which can be accessed remotely across application domains in same machine or different machines.

2.3.5 Summary of Component-Oriented Technologies

The table 2.2 shows the comparison between component-oriented technologies. These component oriented are compare through three aspects, there are development environment, compatibility and portability and implementation. Development environment is used to justify which environment is available to develop the system using there technologies. The compatibility and portability is selected because based on there aspect the domain of the system is identify. Based on comparison form implementation aspect, it justify which technologies is available implement for which area.

PECOS can be developing in composition environment, COM available in application programming interface while CORBA available in platform independent environment and .NET is specific for web-based environment. From the aspect of compatibility and portability PECOS is strong in embedded system, COM is strong in network-based component interaction, CORBA is strong in client and server component and .NET is strong in web-service application. PECOS could be implemented in embedded systems, COM is implemented in windows platform, while CORBA is available implemented in Object request broker product and .NET is available in XML web service.

Table 2.2: Comparison of Component-oriented technologies

Component technologies	PECOS	COM	CORBA	.NET
Development environment	Composition environment	Application Programming interface	Platform independent	Web-based environment
Compatibility & portability	Strong in embedded system	Strong in network-based component interaction	Strong in client and server component	Portable in web-service application
Implementation	Field device	Windows platform	Object request broker product	XML web service

2.4 Graphical Programming

Graphical programming involves block-based code development and offers a more intuitive approach to designing system. A typical graphical code consists of various blocks interconnected by wires. The blocks which might consist of other sub-blocks are the processing units and the wires are responsible for transferring data from one block to another. Graphical programming is based on the concept of data flow, this means the execution of a block or a graphical component is dependent on the flow of data, or more specifically a block executes when data are made available at all of its inputs, and output data of the block are sent to all other connected blocks. Data flow programming allows multiple blocks to be run simultaneously since their

executions are determined by the flow of data and not by sequential lines of code, which is the case in text-based programming (Kehtrnavaz and Gope, 2006).

2.4.1 LabView

LabVIEW are short for Laboratory Virtual Instrumentation Engineering Workbench is a platform and development environment for a visual programming language from National Instruments. LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, various flavors of UNIX, Linux, and Mac OS. The original implementation of LabView was focused at providing a graphical tool for measurement tasks in the area of laboratory automation. According to Jamal and Wenzel (1995), the LabView viewed as a hierarchy of instrument like models called Virtual Instruments (VIs). The Vis present an interactive front panel control, that provide input and display output and an interior block diagram that constitutes the instrument's functionality.

LabView consists of two major components which are Front Panel (FP) and Block Diagram (BD), FP provides the graphical user interface while BD contains the building block of the system and resembles of flowchart (Kehtrnavaz and Gope, 2006). The LabVIEW programming environment provides examples and the documentation, which allows non-programmers to build programs by simply dragging and dropping virtual representations of the lab equipment with which they are already familiar and makes it simpler to create small applications. Other than that, LabVIEW also supports accessing instrumentation hardware.

LabVIEW also has ability in control applications, which can be run on Linux-based workstation, windows-based and embedded controllers running a real-time operating system. On Windows LabVIEW is used to control relatively slow processes which do not require fast update rates, for example, temperature control systems, as Windows is a non-deterministic operating system. On the desktop LabVIEW is used to provide the graphical user interface for embedded systems. On embedded real-time systems LabVIEW development environment is used to deliver deterministic and improve real-time performance, which enabled development of high-speed, closed-loop control applications with the ease-of-use of graphical programming.

2.4.2 UML-RT

UML-RT is an approach modeling for embedded real-time software development. UML-RT is an extension of the standard Unified Modeling Language specifically for modeling embedded or real-time systems, by using the extensibility of UML. UML-RT uses the basic UML mechanisms of stereotypes and tagged values for defining three new constructs; capsule, protocol and connector (Moller et al., 2004).

Capsules describe possibly complex active classes that interact with an environment through input and output signals. Each capsule is associated a unique behavior, which given by a state machine. The protocols define the possible way a capsule can interact with its environment, offering a set of input and output signals, like services of an interface. The actual interaction occurs through ports, which are declared in the respective capsules. Ports are instances of protocols and can control the flow of information in communication of capsules. Ports can be public or private,

2.4.3 Simulink

Simulink is a tool for modeling, simulating and analyzing multi-domain dynamic systems which were developed by the MathWorks. The primary interface of Simulink is a graphical block diagramming tool and a customizable set of block libraries. The Simulink is widely used in control theory and digital signal processing for simulation and design. Simulink extends MATLAB to include many features specific to the modeling and simulation of dynamic systems and also has all of the same system requirements as MATLAB. Simulink provides a graphical user interface (GUI) that is used in building block diagrams, performing simulations, as well as analyzing results.

The Simulink models can be an architectural level designs of software systems and the simulation facilities allow such models to be executed and observed, this property of Simulink can be carry out an advantage for effective dynamic testing. Simulink has a block called models and in simulink the code implementation and input/output entities are not distinguished explicitly (Kehtrnavaz and Gope, 2006). Each blocks in simulink connected by lines and implements the function on its input and output result, which output of blocks from inputs to other block. The following figure is an illustration of a simple Simulink model.

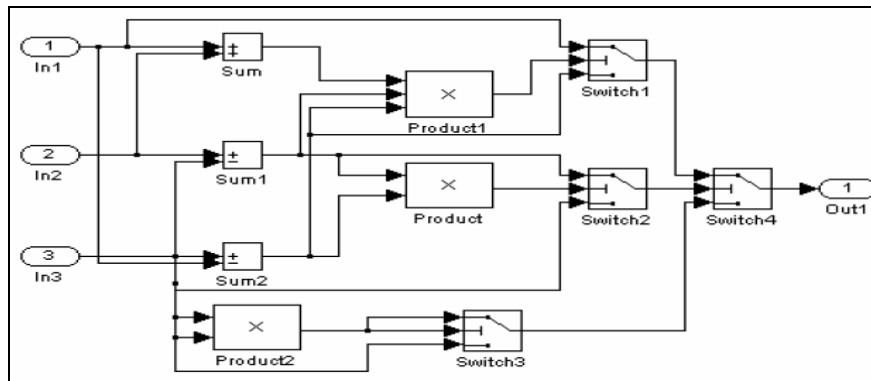


Figure 2.11: Simulink Original Model [Zhan & Clark, 2005]

2.4.4 Summary of Graphical Programming

The graphical programming that has been reviewed in this study is LabVIEW, UML-RT and Simulink, and the summaries show at table 2.3 at below. The graphical programming tool is compared from five aspects, which are pattern-oriented, component-oriented, analysis and design, language and platform. The pattern-oriented and component oriented are used to compare there tools because based on there two aspect, the tools which based on pattern and component are identify. The analysis and design aspect is used to justify either there tools cover on analysis and design phase. The language and platform aspect are used to identify, which language and platform are available to implement there tools.

These graphical programming are based on component, while the pattern-oriented is not used in tools. The UML-RT and Simulink is on analysis and design level, except LabVIEW. There tools has they own specific language and platform, for example LabVIEW is specific language for visual programming and available at platform such as Microsoft Windows, UNIX, Linux and Mac OS, while UML-RT is Unified modeling language and available for embedded real-time platform, and

Simulink is specific language for diagramming and available at simulation application.

Table 2.3: The comparison of Graphical programming

Graphical programming	LabVIEW	UML-RT	Simulink
Pattern-oriented	No	Yes	No
Component-oriented	Yes	Yes	Yes
Analysis & design	No	Yes	Yes
Language	Visual programming language	Unified Modeling language	Diagramming language
platform	Microsoft windows, UNIX, Linux & Mac OS	Embedded Real-time	Simulation application

2.5 Software Process

Software process is a process or a set of processes used by an organization or project to plan, manage, execute, monitor, control and improve its software related activities. The scope of software process is larger than that of software engineering, which software process concern on activities like system and software requirement analysis, software design, software implementation, testing and maintenance (Lepasaar and Makinen, 2002). A process consists of activity, which cover software development and maintenance activities such as project management or quality








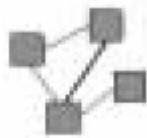
assurance activities. Capability Maturity Model Integration (CMMI) and Software Process Engineering Metamodel (SPEM) are example of software process.

2.5.1 Software Process Metamodel Engineering (SPEM)

Software process Metamodel Engineering (SPEM) is a meta-model that used to describe a concrete software development process or family of related software development process. SPEM is tools which were develop by OMG SPEM used object-oriented approach to modeling a family of related software processes and use UML as a notation. SPEM is a meta-model for defining processes and their component. SPEM is an appropriate approach for the software process engineering domain.

SPEM consist with stereotypes, there are 1) WorkProduct, 2) WorkDefinition, 3) activity 4) actor and 5) phase. Table 2.4 shows the stereotypes of SPEM.

Table 2.4: Stereotypes of SPEM

WorkProduct	
WorkDefinition	
Activity	
Actor	
phase	
ProcessPackage	
Document	
UMLModel	

SPEM is selected to be used in this project, because the approach of SPEM used UML as a notation which related with UML-RT tool that going to be used to design the software process for integrated of POAD and PECOS.

2.6 Summary

Based on the comparison at table 2.1, table 2.2 and table 2.3, the pattern methodology used in this project is POAD, while the component-oriented technology used is PECOS and graphical programming selected is UML-RT. POAD is chosen because POAD provides the structural approach, while PECOS as COP is selected because PECOS supports component-based software development for embedded system and UML-RT is selected because this tool is a specific tool for embedded real-time system and can support to achieve the objectives that has been define at chapter 1. In order to define the software process into formal form, the Software Engineering Meta Model (SPEM) is selected.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter discusses the methodology and the framework that will be used in this project. Methodology is the specific way of performing an operation or sequence of methodical work through steps, which is used as a guideline to implement the project, in order to accomplish the objectives.

3.2 Operational Framework

This section is explains the detail about the prime activities which has been carried out in this project to achieve the objectives. The goal of this section is to sectionalize the work in order to help in developing the project schedule. The

following figure shows the operational framework stages, which is used in this project.

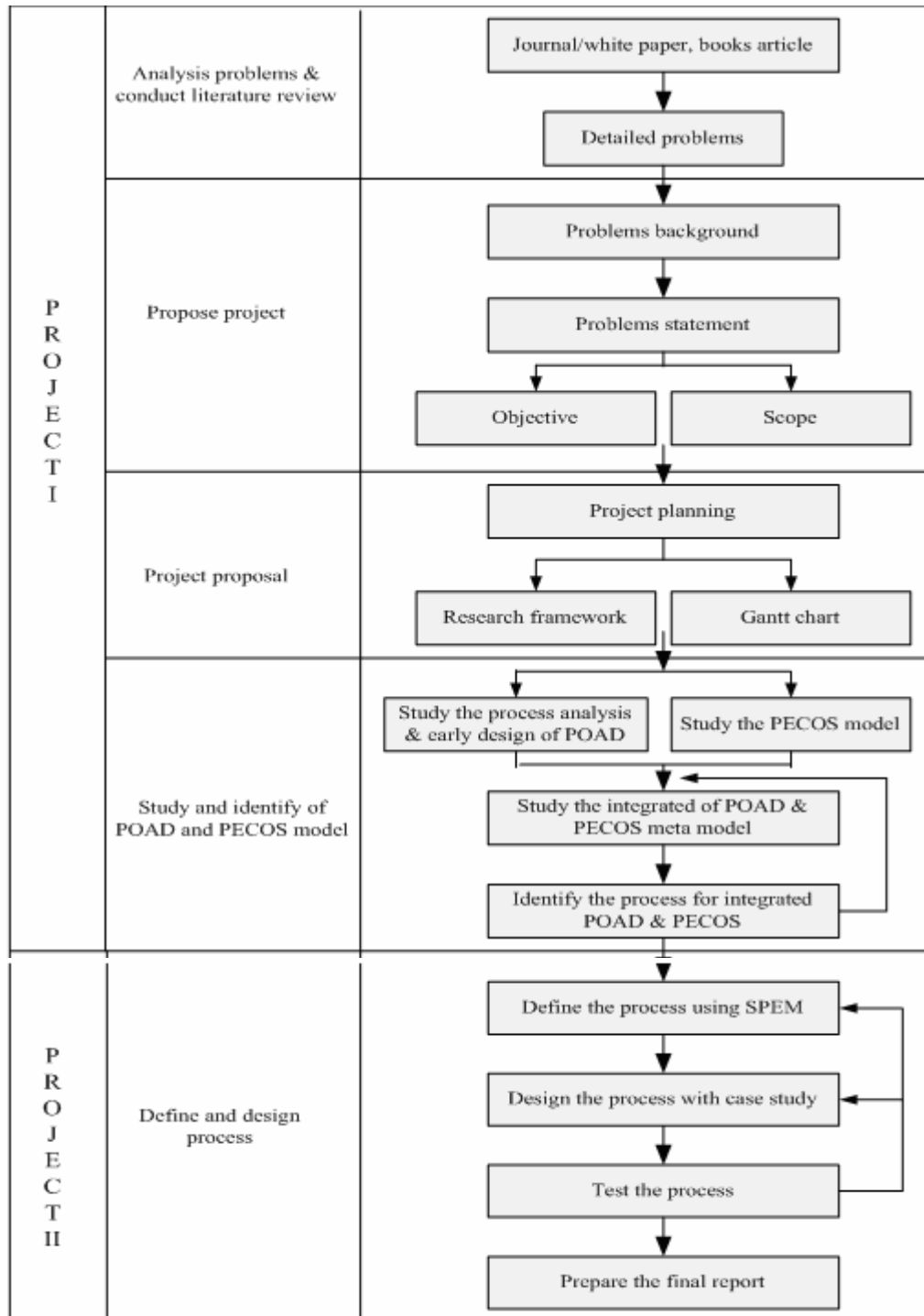


Figure 3.1: Operational Framework

3.3.1 Analysis problems and conduct literature review

Analysis problems and conduct literature review is the first phase on this project. Firstly, before proceed with analysis problems, we need to find and study some journals or book article which is related with the topic. Based on the study, the problems in area of interest can be defined and then the analysis on those problems can be continued. In this project the area of interest is pattern-oriented and component-oriented on embedded real-time system. The main goal of this project is to identify and define the software process of integrated POAD and PECOS into the formal form on embedded real-time systems.

The next steps are conducting literature review. Literature review is the detailed study about elements in component-oriented technology, pattern-oriented methodology, graphical modeling and software process. Basically the literatures are reviewed from books, soft copy and hard copy of journal. Reviewing in the early stages were focus on WHAT things mean, like definitions and explanations of POAD and PECOS. The second is HOW things can work like the process structure. Based on the literature review the detailed problems can be obtained which is can contribute to propose the project.

3.3.2 Propose Project

Propose project is the second phase in this project. Based on the study about the four elements, there are pattern-oriented, component-oriented, graphical modelings and software, the understanding about what and how can be achieved. The detailed problems gained can be contributed in defining the problem background,

problem statement, objectives of the project and scopes. These elements can formulate to contribute project proposal.

3.3.3 Project Planning

Project proposal contains problem background, problem statement, objectives and the scopes of the project, where these entire element, will be documented and submitted to project committee to be approved. After the project proposal approved by project committee, project planning needs to done carefully in order to make sure the project will finish on time. Normally, project planning is contains plan how to go further on that project such as come out with research framework and Gantt chart. Based on research framework and Gantt chart the project can be continued successfully.

3.3.4 Identify and study POAD and PECOS

In order to identify and define the software process of integrated POAD and PECOS, it is important to study and analyze the behavior or characteristic of POAD and PECOS. Moreover, the analysis and the understanding of the combined of Meta model consists of POAD and PECOS, which has been done by Jawawi (2005) is very important. Based on understanding of this combination, an understanding on how to identify the software process of integrated POAD and PECOS can be sustained.

The defining of software process of integrated POAD and PECOS will be continued in project 2. Based on the software process which has been identified during project 1, the process will be defined into formal form using SPEM. After defining the process into formal form, the process for integration of POAD and PECOS will be developed, the suggested methodology; UML-RT is used to show the process for integration between POAD and PECOS. After development of process is done, then the developer needs to test the process with a case study. The testing activity is important to ensure the quality of software such as complexity, maintainability can be achieved. The last activity in this operational framework is preparing the final report, after the testing activity is complete then, the next steps is doing the report and submit into committee.

3.4 Hardware and Software Requirement

In this chapter, hardware and software that will use in this project are listed. This list can helps to make the project process clear and easy and present that project is feasible to be carried out.

Works are carried out using a laptop:

Laptop Brand Dell with processor centrino, as a main use of researcher's work, such as doing analysis and report.

Software that would be used to carry out the project are:

Microsoft Windows XP Professional

Microsoft Office 2007

Adobe Acrobat Professional 6.0

Microsoft Visio 2003

Microsoft Visual C++ 6.0

Rational Rose Real Time 2003

3.5 Project Schedule

The project schedule is contains the activity of project and the date. In order to make project schedule is clear the Microsoft project is use to shown project activity. The Figure of project schedule for project 1 show at Appendix A, while schedule for project II shows at Appendix B.

3.6 Autonomous Mobile Robot (AMR) Case Study

Case study that are going to be use in order to test the software process for integrated of POAD and PECOS is Autonomous Mobile Robot (AMR). Autonomous Mobile Robot (AMR) represents a mechatronics system, which involves expertise from multi-disciplines in the domains of artificial intelligence, mechanical, electronics, computer and software engineering to develop it. An AMR system is a class of ERT system with many possible applications and market. The autonomous system of the AMR is capable of traversing a terrain, performs its design tasks, sense its environment and intelligently react to it. The technologies in the AMR include commercial mobile personal robots and service robot.

APIBOT is a small wheeled AMR with an approximate diameter of 16 cm and consists of an aluminium body and a pair of wheels. Each drive wheel is actuated by a servo motor. The central processor on APIBOT is an ATMEGA32 8-bit single chip microcontroller from ATMEL. The ATMEGA32 microcontroller has on-chip 32 kilobytes of ROM for program storage and 2 kilobytes of RAM for data and stack usages. On the ATMEGA32 chip, there are also eight channels of analog-digital-converter which can be used for reading sensor outputs and four-width modulation (PWM) outputs for control of servo motors. The block diagram to show the devices controller by microcontroller is illustrated in Figure 3.2.

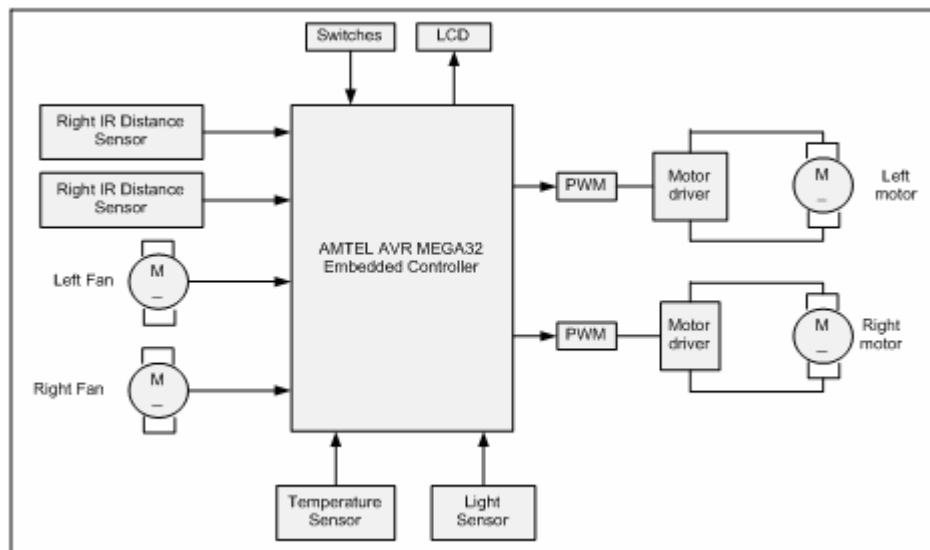


Figure 3.2: Block Diagram of the APIBOT

The main task of the robot software on APIBOT is to navigate the robot intelligently in a room while avoiding obstacles trying to locate source of fire and automatically extinguishes the fire. To assist APIBOT on this task, on-board there is Sharp GP2D12 Infra-Red (IR) distance measuring sensor used for obstacle avoidance, two light sensors and temperature sensor for fire detection, a battery sensor for on-board battery monitoring and two small direct-current (DC) motors actuating two fans acting as fire extinguishers.

The embedded software must support the intelligence aspect of the robot in order to respond to the conditions in the environment in achieving the goal. The intelligence of AMR is supported by a behavior-based control using subsumption architecture. To support multi-tasking requirements of the software, a pre-emptive Real-Time Operating Systems (RTOS) is used in the robot software. The embedded controller also communicates with human through Liquid Crystal Display (LCD) and switches. To complete software system shall be embedded on AMR, and all processing must be carried out on-board.

3.7 Summary

This chapter discusses the methodology that will be used in order to fulfill the objectives of the project. The operational framework presents the steps, which guides the researcher to achieve the main goal of this project. Based on the methodology, the objectives that have been defined in Chapter 1 can be achieved.

CHAPTER 4

POAD AND PECOS PROCESS MODEL

4.1 Introduction

This chapter discusses about the software process for integrated POAD and PECOS. The process is presented using Software Process Engineering Meta-model (SPEM).

4.2 The Software Process Engineering Meta-Model (SPEM)

Software Process Engineering Meta-model (SPEM) is a meta-model that is used to describe a concrete software development process or family of related software development process. SPEM is a meta-model for defining processes and their component. SPEM is an appropriate approach for the software process

engineering domain. The Figure 4.1 below shows the SPEM icons that are used in this project:

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

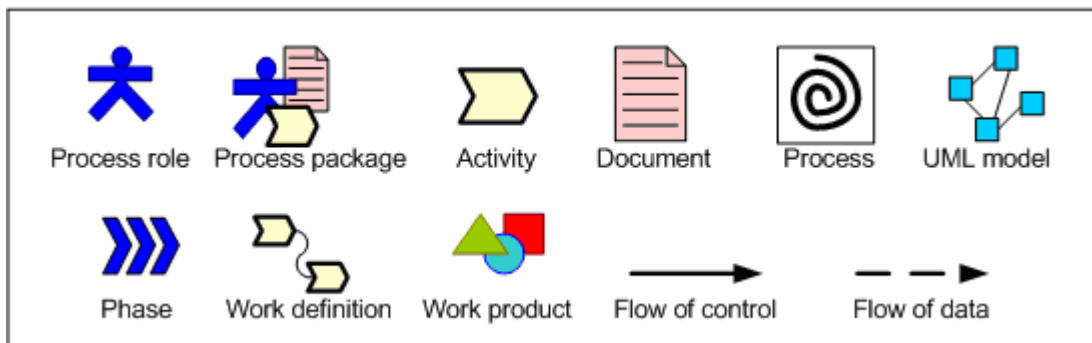


Figure 4.1: SPEM Icons

4.3 The Process Model

The defining process POAD and PECOS from the existing Meta model into formal form is important to enable and support the development of CASE tool for ERT system. Based on the process integrated of POAD and PECOS Meta model, there are three phases 1) analysis phase 2) early design phase and 3) detail design phase. Each of these phases will produce work product for the related activities and have own responsibilities for designing one or more artifact.

The purpose of the analysis phase is to produce a set of component by capturing the requirements of an application. The phase includes four activities that are capturing the requirement, identify the component, select the pattern and construct the pattern level diagram. The aim of the early design phase is to transform the analysis model to design model by analyzing the relationship between the instantiated components. The early design phase contain two main activities that are constructs pattern level diagram with interface and construct pattern level diagram with control interface. The aim of the detail design phase is to produce a detail model design of software. The activity at detail design phase include identify the design component, construct design component, refinement of design component and construct detail component level design. The Figures 4.2 at below illustrate the disciplines of POAD and PECOS process.

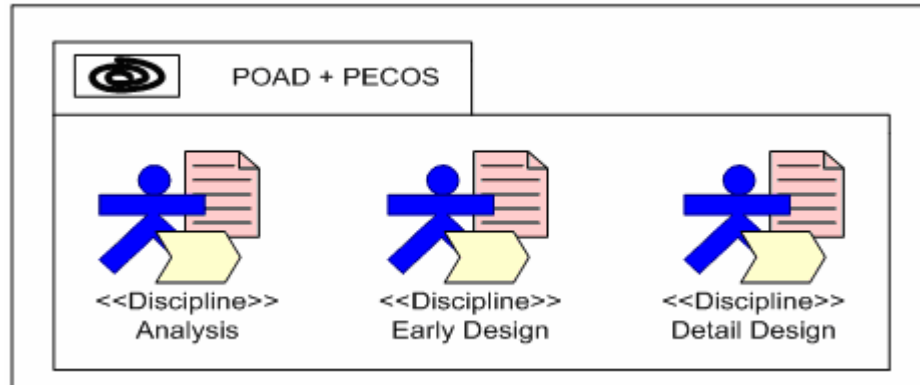


Figure 4.2: The disciplines of POAD and PECOS

The Figures 4.3 below illustrate the phases in POAD and PECOS process. Based on existing POAD and PECOS Meta model, the analysis phase and early design phase activity used POAD methodology, while detail design phase is used PECOS Meta model. The analysis model is produced during analysis phase, while the early design phase produces the design model and during detail design phase activity the detail design model is produced.

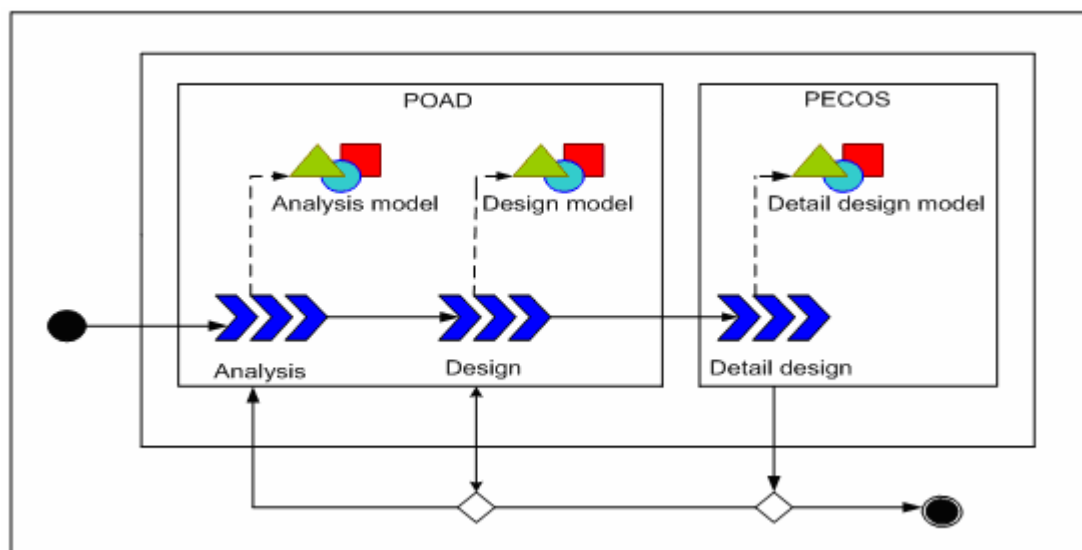


Figure 4.3: Phases in POAD + PECOS process

4.3.1 Use Case diagram

Use-case diagram is used to show the relationships between process roles and the main activities in software process. Three use-case diagrams were developed for analysis, early design and detailed design in software process for integrated POAD and PECOS, such as shows in Figure 4.4, 4.5 and 4.6.

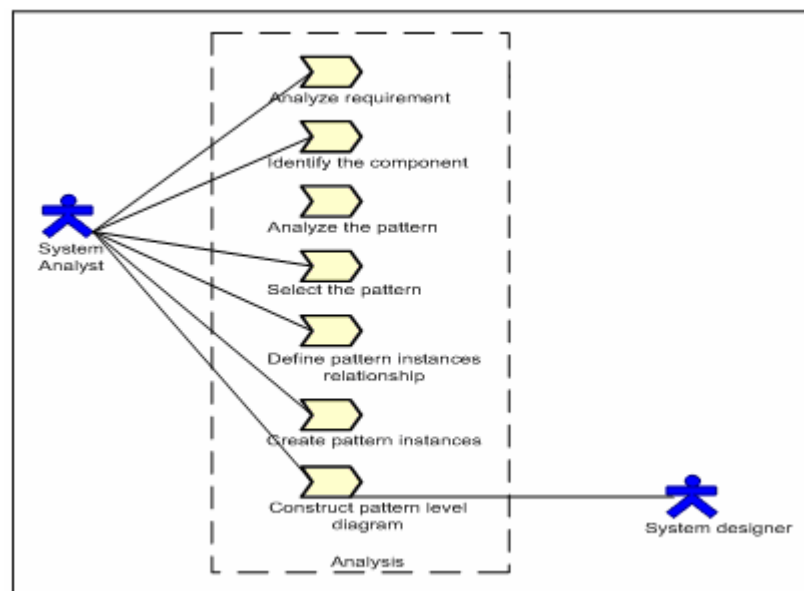


Figure 4.4: Use case diagram for analysis phase

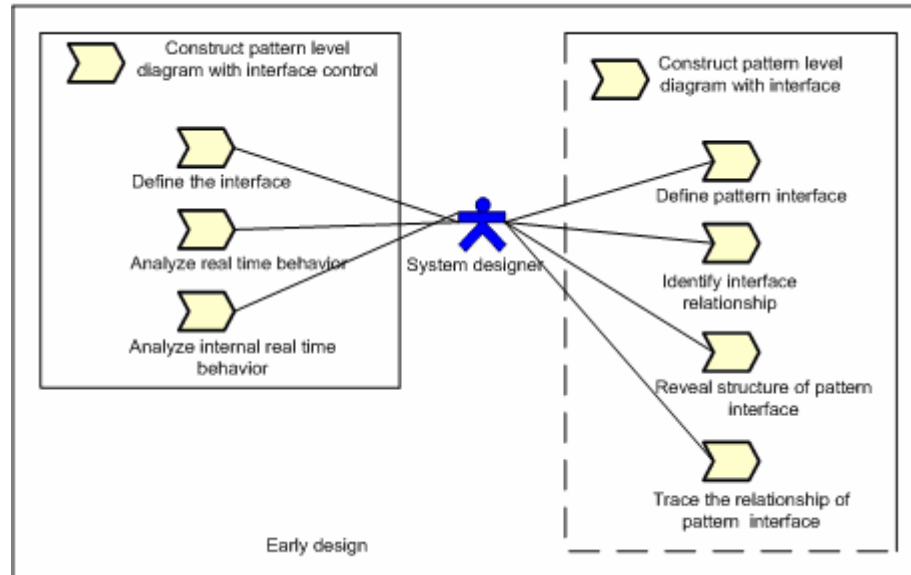


Figure 4.5: Use case diagram for early design phase

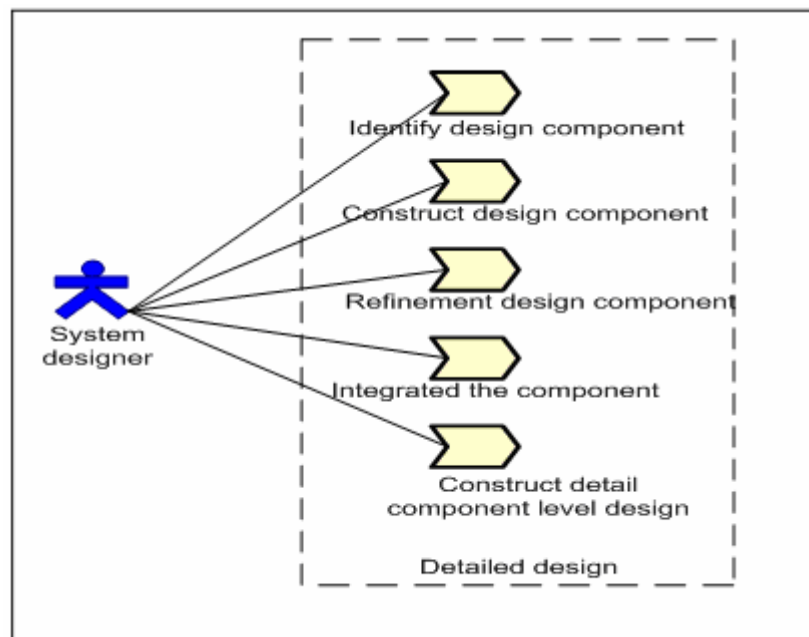


Figure 4.6: Use case diagram for detailed design phase

As shown in Figure 4.4, 4.5 and 4.6, there are two process role that involve in POAD and PECOS process; system analyst and system designer. Each process role is

responsible for the work product produced by his activities. System analyst involve in the activities related to analyze requirement, identify the component, analyze and select the pattern, create pattern instances, define pattern instances relationships and construct pattern-level diagram. The system designer at the analysis phase is an individual who use the pattern-level diagram as an input in she or he work.

System designer at early design phase involve in the activities related with two main activities, there are construct pattern level diagram with interface and construct pattern level diagram with control interfaces. The activity involve in construct pattern-level diagram with interfaces are defines pattern interfaces, identify pattern interfaces relationships, reveal structure of pattern interface and trace the relationships between the pattern interface. The activities in construct pattern-level diagram with control interface involves define the interface, analyze the real-time behavior and analyze the internal real-time behavior. System designer at detailed design phase involve in activities related with identify design component, construct design component, refinement design component, integrated the component and construct detail component level design.

4.3.2 The Analysis Phase

The purposes of analysis phase are to analyze the application requirement, identify the set of component, select the pattern and construct the pattern-level diagram. The analysis phase will produce two work product (component and pattern) and one model called pattern-level diagram. Figure 4.7 illustrates the exploitation of the analysis phase represented as a SPEM discipline.

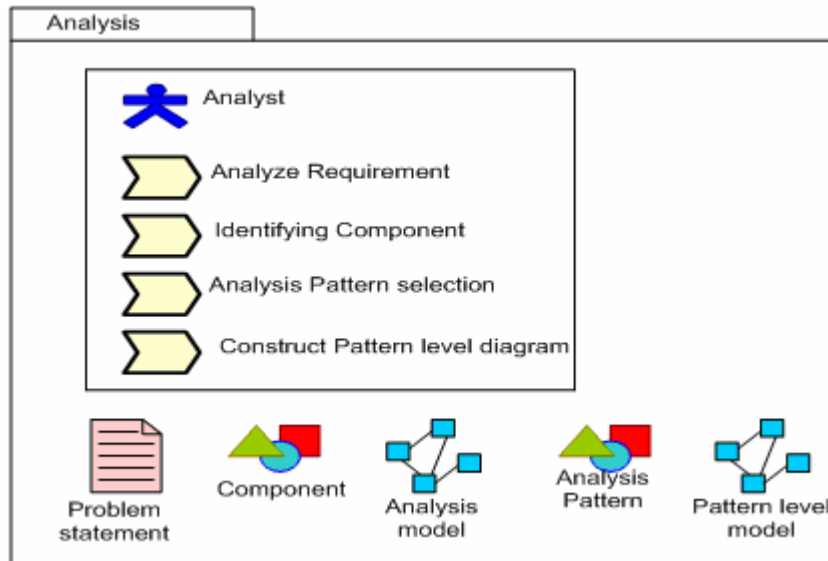


Figure 4.7: POAD + PECOS Analysis phase

The analysis phase contains two main activities that are problem analysis and integration of analysis component. The problem analysis activities analyze the requirement and matching the requirement from user with problem domain. The analysis phase start with analyzing the requirement by identifying the problem, looking at the application functionalities that should be provided and identify the user who interact with application. Based on application requirement as input, the next step is model it in the UML use case diagram (analysis model). The use case diagram shows the main use case that captures the required functionalities, their relationships and the interactions of the actor.

The modeling of the analysis model is performed in an activity called integration of analysis component. The integration of analysis component contains activities such as identify the component, select the pattern and construct the pattern level diagram. Based on analysis model which were modeled, the component will be identified by mapping the use cases with the context and problem elements available in the application. Now the process of selecting the suitable pattern to model the system is conducted, this process is called analysis pattern. The analysis pattern will

be achieved by creating the pattern instances, identifying and defining the pattern instances relationship.

The next step in the analysis phase is constructing the pattern level diagram. Based on the pattern instances of the selected analysis pattern and the relationship between these instances, the pattern-level diagram will be produced. Figure 4.8 shows the flow of activity in the analysis phase, which is described in terms of work definition and work product.

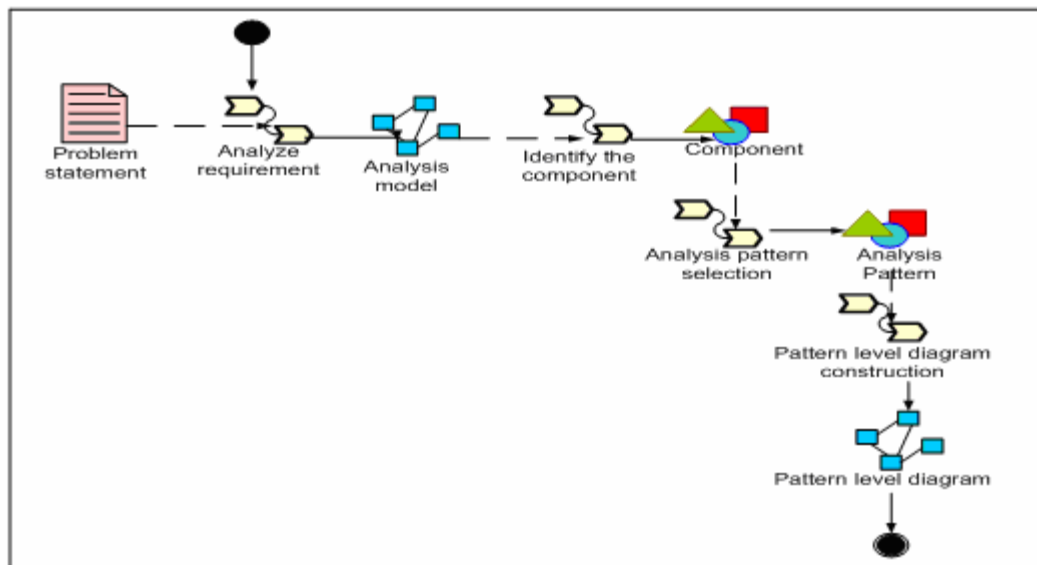


Figure 4.8: The Analysis phase describe in terms of works definition and works product

4.3.3 The Early Design Phase

The purpose of the early design phase is to transform the analysis model into a design model by analyzing the relationship between the instantiated patterns or components using pattern interfaces. The pattern level diagram at the analysis phase is

used as input for early design. The additional function will be added into pattern level diagram by providing the interfaces and control for it interfaces.

The task at early design phase includes define pattern interface, identify interface relationship, reveal structure of pattern interface, trace the relationship between pattern instances and analyze the real time behavior. There are two models that will be produced at early design stage; pattern level diagram with interface and pattern level diagram with control interface. The Figure 4.9 at below illustrates the early design phase described in term of works definition and work product.

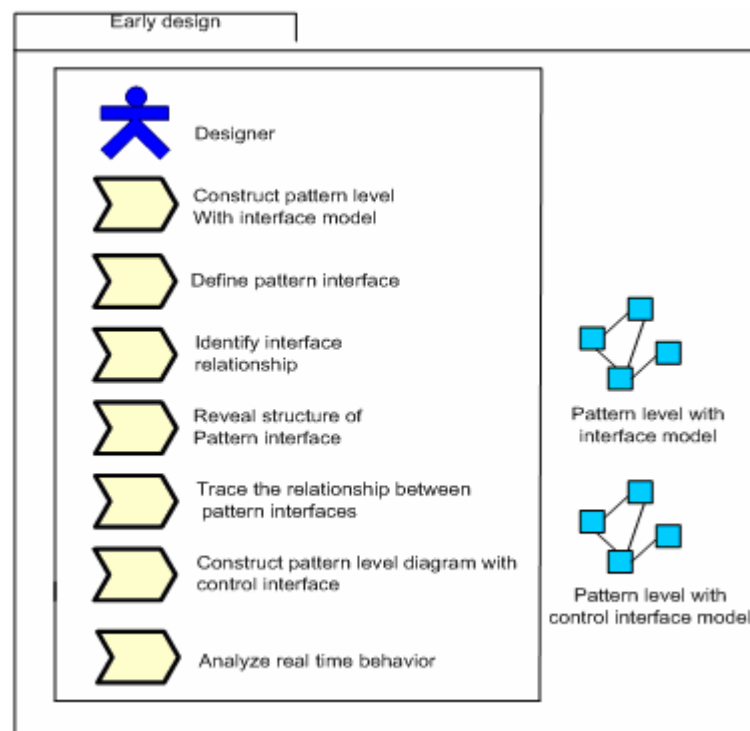


Figure 4.9: POAD + PECOS Early design phase

The early design phase has two main activities there are pattern-level diagram with interface construction and pattern level diagram with control interface construction. The figure 4.10 at below show the steps at early design phase:

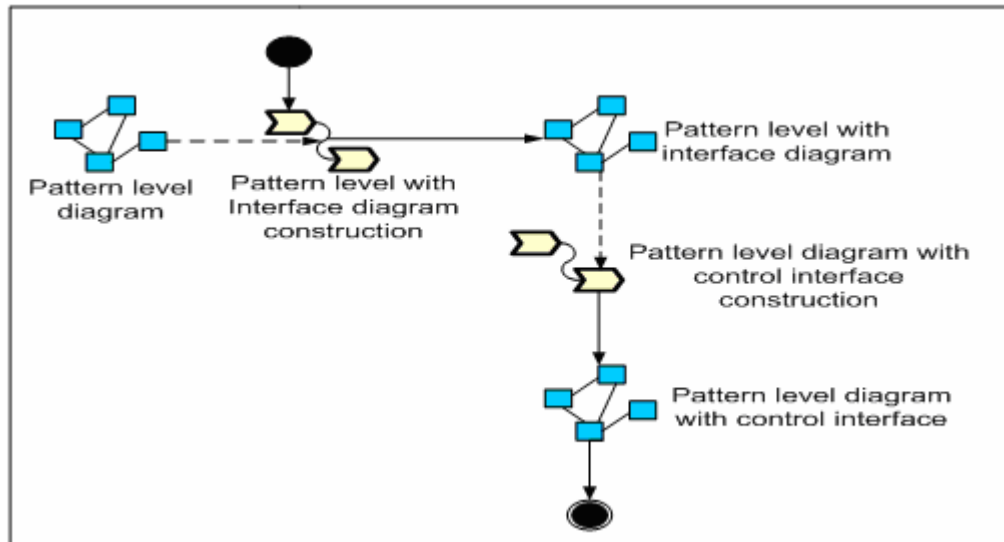


Figure 4.10: The Early design phase described in terms of works definition and work product

4.3.3.1 Pattern level diagram with interface construction

The purpose of this activity is to create the pattern-level with interface diagram for the application. At the pattern level diagram with interface the relationship between pattern instances will be detailed out to lower-level design relationships using interface. The creation of pattern level diagram with interface can be achieved by adding the pattern level diagram with interface. The interfaces are means by which components connect. Once the interfaces of the model are created, the relationships between the interfaces need to be defined and the pattern interfaces should be traced.

4.3.3.2 Pattern level diagram with control interface construction

The purpose of this activity is to create the pattern-level diagram with control interface. The pattern-level diagram with control interface was produced by adding the control for the interfaces, which belong to active and events class. The control interface can be done by analyzing the internal real-time behavior. The control interface will define the interface to support the real-time temporal behavior and the synchronization services between packages. The control interface defines the attributes of the real-time requirements of a pattern component and this is necessary in active and events classes.

4.3.4 The Detailed Design Phase

The purpose of this phase is to create a detailed model of the software. In order to produce a detail design model for each component, the necessary information is needed by the early design model. Based on Figure 4.2 above, the detail design phase is developed using PECOS. The migration from POAD to PECOS can only be performed after specification of a pattern as a design component provided by the interfaces. POAD uses class and operation as interface in PatternInterface element, while PECOS uses Port as interface component. This element will be mapped to enable PECOS model usage at the detail design phase.

The detailing of real-time behavior is another element. It enables detail design developed using the PECOS model. The real-time behavior which was introduced in POAD methodology can be transformed directly to the relevant component type in

PECOS. The real-time behavior at early design phase represented by controlInterface element in POAD can be translated and detailed into property element in PECOS.

The Figure 4.11 illustrates the detail design phase in terms of works definition and work product. Based on that figure, the activity at detail design phase includes identify design component, design component construction, refinement design component and construct detail component level design. There are two work products that will produced during activity at detail design phase; design component and detail component level design.

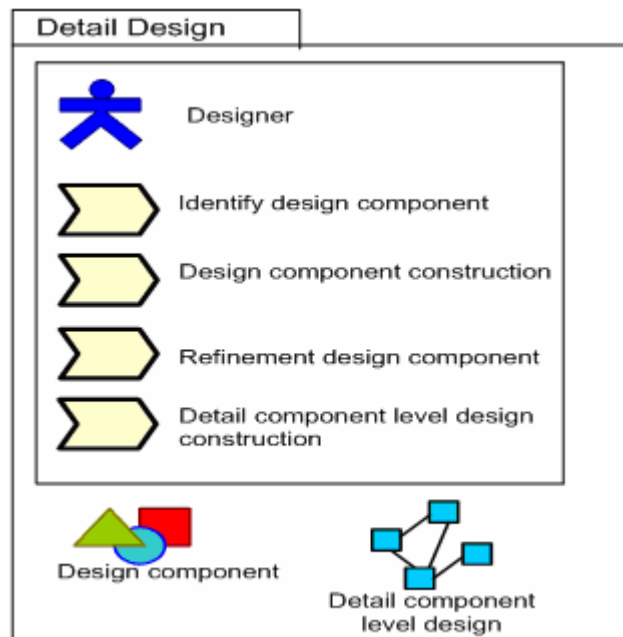


Figure 4.11: The detailed design phase in terms of works definition and work product

The Figure 4.12 illustrates each step at detailed design phase. The detailed design phase start by identifying design component of the design model at early design phase that is used as an input. The next step is construct design component. In

this activity pattern level diagram with interface and pattern level diagram with control, interfaces are used as an input.

The construction design component can be done by matching the solution in the design with the solution from reusable design. The refinement design component can be done by elaborating and grouping component into new components design model if the component cannot be reacquired from the reusable design component. The product from this step is design component. Based on diagram at early design phase and design component as an input, the construction of detail component level design can be proceeding. The selected component solution will be integrated in order to construct the detailed component level design. The product that will produce based on their activity is called detailed component level design.

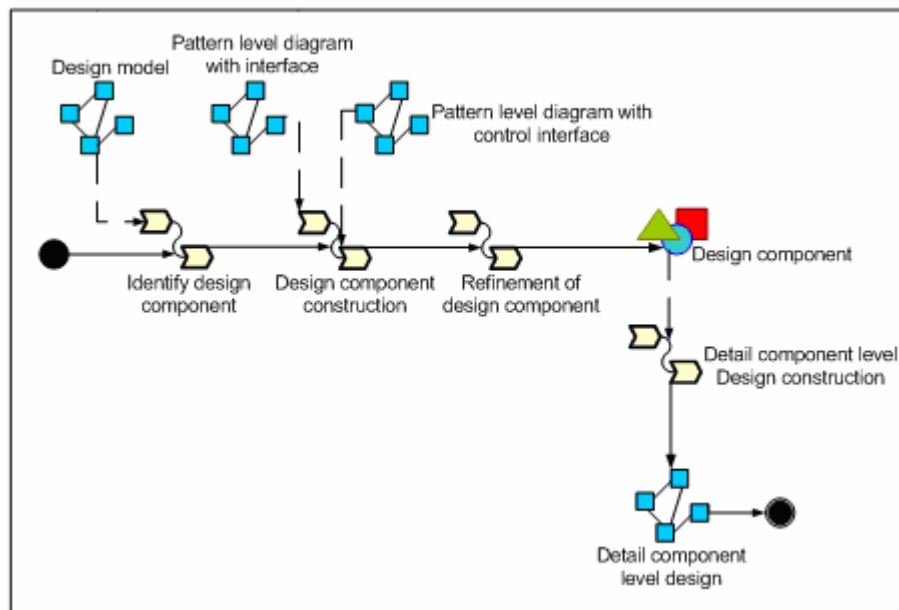


Figure 4.12: The activity of detail design phase

4.4 Discussion on Process Model

This chapter describes the integrated POAD and PECOS process and defines the process using SPEM icons. The process has three phases: analysis phase, early design phase and detail design phase. The analysis phase and early design phase are developed using POAD methodology, while detailed design phase used PECOS model. Each of these phases will produce a work product. During these activities, the analysis phase product called pattern level diagram is used as an input to proceed next phase.

During the early design phase the work product are pattern level diagram with interface and pattern level diagram with control interface. These two models will be used to create detailed component level design. The detailed component level design is product from detailed design phase.

The flow of process or activity in each phases is clear by defined the process model using SPEM icons. Therefore, the defining process using SPEM can be support software engineer to estimate their tasks and improve the development of software. Moreover, the artifact during each activity is defined clearly, which can support the documentation task during software development process. However, SPEM only define the structure of integrated POAD and PECOS, because the integration POAD and PECOS cannot express the functional behavior of the system.

CHAPTER 5

THE PROCESS MODEL USING UML-RT

5.1 Introduction

In order to implement process model for integrated POAD and PECOS, the UML-RT tools is apply into that process model. An AMR case study is applied, in order to demonstrate process model using UML-RT tools.

5.2 Mapping Process

In order to design the process model using UML-RT, the mapping process is needed between POAD and UML-RT and UML-RT and PECOS. Based on defining process using SPEM, analysis phase and early design phase are design using POAD method, while the detailed design phase is design using PECOS model. Therefore, the mapping POAD into UML-RT and UML-RT into PECOS are needed.

5.2.1 Mapping POAD and UML-RT

In order to use UML-RT as a tool to design the diagram at analysis and early design phase, the mapping between POAD and UML-RT is needed. The mapping process involve two elements there are structure and behavior, there are three model from analysis and early design phase will be map with UML-RT 1) pattern level diagram, 2) pattern level diagram with interface and 3) pattern level diagram with control interface.

The principle constructs for modeling structure using UML-RT are capsules, port and connector. Capsules are specialization of classes with communication mechanism and consider as a component. The ports are object which can provide interface mechanism to the capsule, while the connector transmits the signal between the ports are connectors. The principles for modeling the behavior using UML-RT are protocol and state machine. The protocols define a contractual agreement between the communicating participant and the state machine will be used to model the behavior of a particular class, protocol and capsule in responding to events message.

Figure 5.1 show the mapping pattern level diagram and UML-RT. The diagram illustrates the capsule in UML-RT map with packages or component in POAD. The association in POAD which illustrate relationships between the packages or component will be map to UML-RT connector.

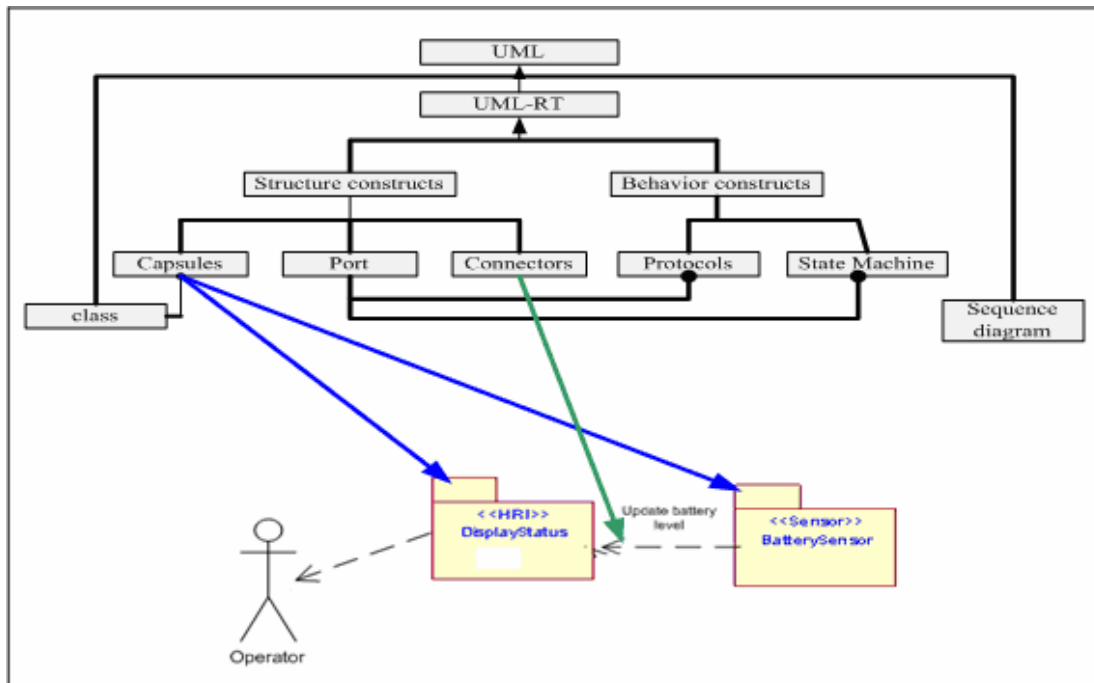


Figure 5.1: Mapping Pattern Level diagram into UML-RT

Figure 5.2 shows the mapping pattern level diagram with interface and UML-RT. The Figure shows the package or component in POAD mapping with Capsule in UML-RT. The interfaces in POAD model mapping with two elements in UML-RT, where interfaces map with Port in structure construct in UML-RT and map with Protocol in behavior construct in UML-RT. The connector in UML-RT is mapping with dependency relationship between the packages or components in POAD model.

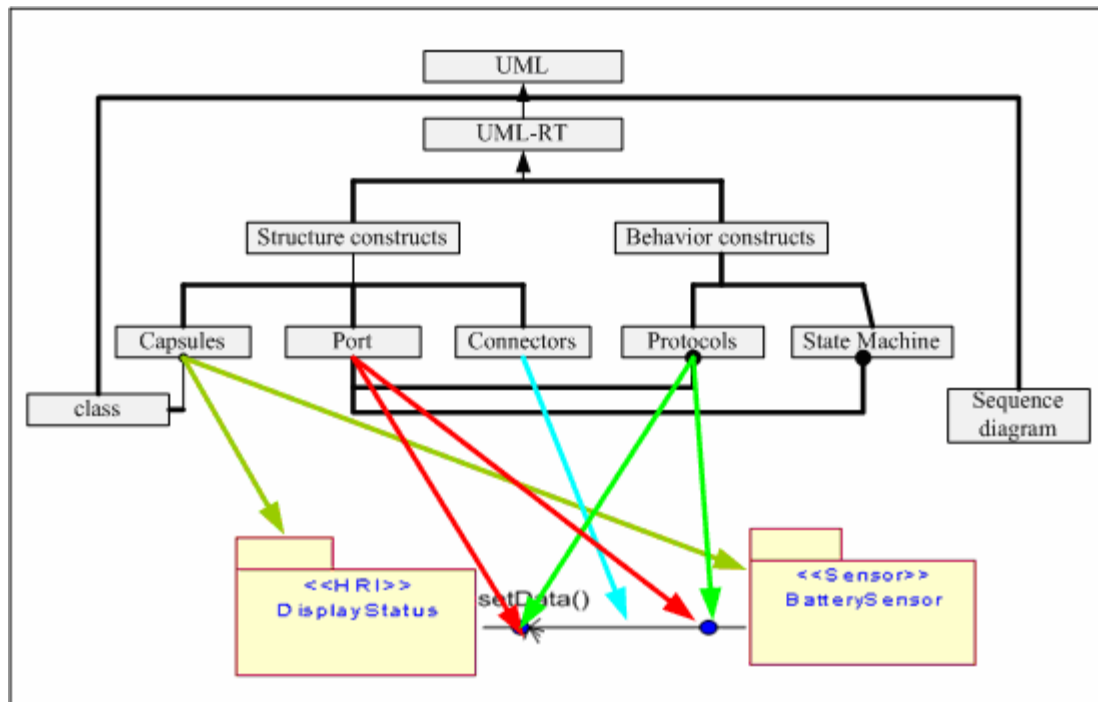


Figure 5.2: Mapping Pattern Level diagram with Interface into UML-RT

Figure 5.3 shows the mapping pattern level diagram with control interface. The figure illustrates the mapping packages or component in POAD model with Capsule in UML-RT, while the dependency relationship between the packages or components is mapping with Connector in UML-RT. The control interface in POAD model is mapping with sequence diagram in UML-RT, which the sequence diagram contain the timing behavior.

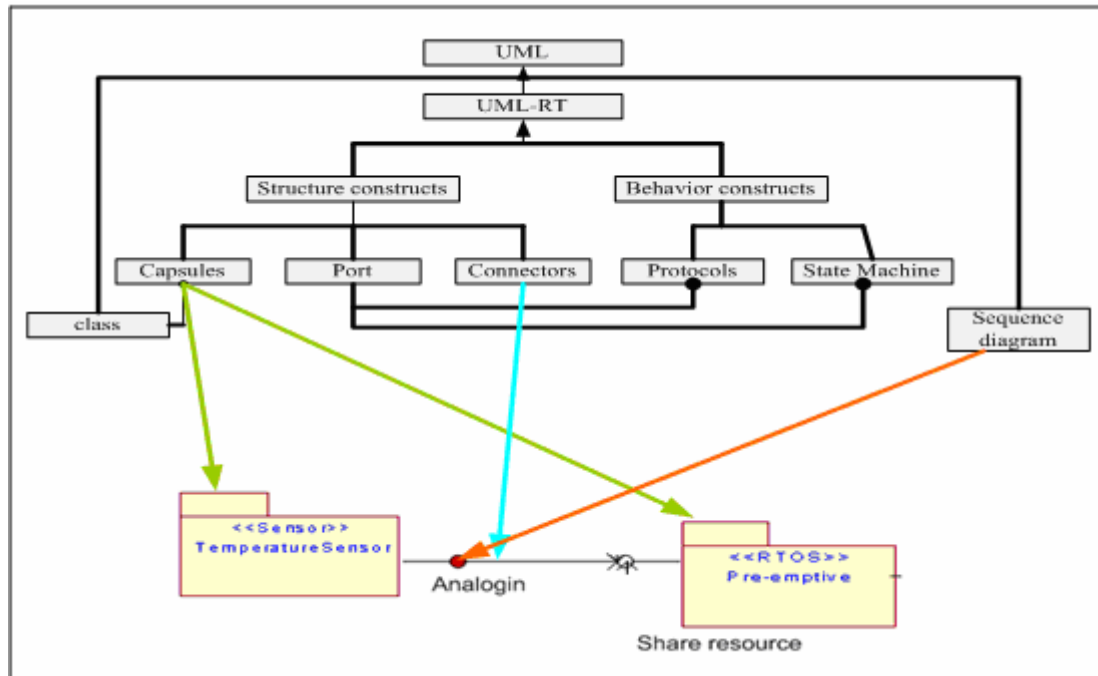


Figure 5.3: Mapping Pattern Level diagram with Control Interface into UML-RT

5.2.2 Mapping UML-RT and PECOS Model

In order to use PECOS model in detailed design phase the mapping process from UML-RT to PECOS model is needed. From the integration POAD and PECOS Meta model (2005), the mapping processes between UML-RT into PECOS involve two elements there are structure constructs and behavior constructs.

The principle constructs for modeling structure using UML-RT are capsules, port and connector. Capsules are specialization of classes with communication mechanism and consider as a component. The ports are object which can provide interface mechanism to the capsule, while the connector transmits the signal between the ports are connectors. The principles for modeling the behavior using UML-RT are protocol and state machine. The protocols define a contractual agreement between the

communicating participant and the state machine will be used to model the behavior of a particular class, protocol and capsule in responding to events message.

Figure 5.4 illustrates the mapping between UML-RT notations with PECOS notation. The diagram shows the mapping of UML-RT Capsules to PECOS components, UML-RT Protocol map with PECOS port, UML-RT Connector to PECOS connector and timing requirements in UML-RT sequence diagram to PECOS property bundle.

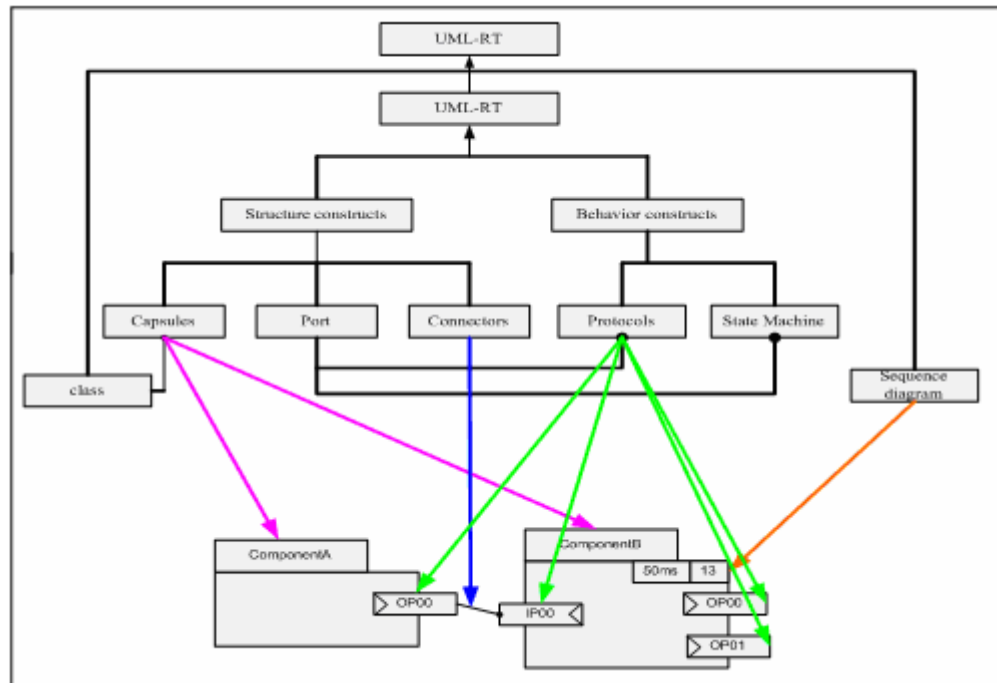


Figure 5.4: Mapping UML-RT and PECOS

5.3 Process Model

The process model for integrated POAD and PECOS consist of three phase, there are analysis phase, early design phase and detail design phase. The AMR application is applied and elaborates according to those phases. The main tasks of the robot software are to navigate the robot in a room while avoiding obstacle trying to locate source of fire and automatically extinguishes the fire. To assist APIBOT on this tasks, on-board Infra-red (IR) distance measuring sensor used for obstacle avoidance, two light sensors and a temperature sensor for fire detection, a battery sensor for on-board battery monitoring and two small direct-current (DC) motors actuating two fans as fire extinguishers. The AMR is supported by a behavior-based control using subsumption architecture. The embedded controller also communicates with human through Liquid Crystal Display (LCD) and switches.

5.3.1 Analysis Phase for AMR

The first step at the analysis phase is analyzing the requirement of the AMR application, and then models it in the UML use case diagram as present in Figure 5.5.

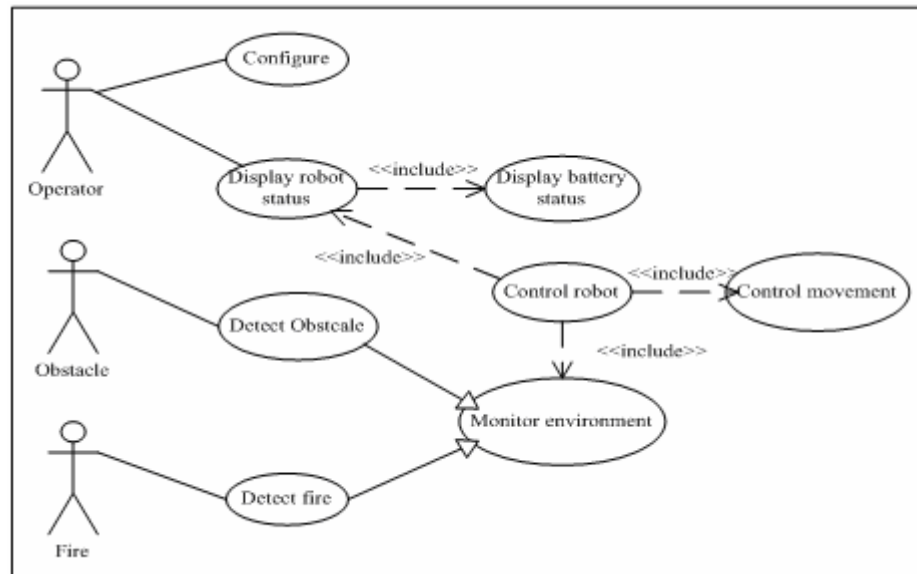


Figure 5.5: The AMR use case diagram

Based on use case diagram at above, three actors are identifying which interaction with the system. The *Operator* who configure and monitor the AMR through LCD display and switches, the *Obstacle* and the *Fire* in the Robot environment, which need to monitor by the AMR. The use cases which identify from the system include: *Configure* use case, *Display robot status* use case, *Display battery status* use case, *Monitor environment* use case, *Detect obstacle* use case, *Detect fire* use case, *Control robot* use case and *Control movement* use case. The *Configure* use case provides the input to the system, the *Display robot status* and *Display battery status* use case which collect the data status and display through LCD. The *Monitor environment*, *Detect obstacle* and *Detect fire* use case which collect the data and update it, while the *Control robot* and *Control movement* use case which control the robot behavior and movement.

The next step at analysis phase is identifying the component and select the component which available with the context and problem elements in the AMR application. After identified the component, the next process is selected the analysis

pattern for AMR application. The analysis pattern consists of two elements there are structure model and real-time behavior. The structure model described the class that make-up a particular component pattern, the combination of class in the structural model is arranged in a package to represent constructional component pattern. The connection between the packages in the analysis pattern is supported by interface, which define in the analysis pattern. The result from component-based analysis pattern is structure model composition by matching the pattern interface from the analysis pattern. The real-time behavior in analysis pattern is used to specify the real-time behavior of real-time components in UML-RT sequence diagram.

By matching the decomposed Use case of Figure 5.5 with the context and problems elements available in AMR analysis pattern, a mapping of the application requirement and AMR analysis pattern is obtained as shown in Table 5.1.

Table 5.1: Mapping Component-based analysis pattern and Use Case

Component-based analysis pattern	Use Case
Sensor Pattern <ul style="list-style-type: none"> • Distance Sensor • Temperature Sensor • Light Sensor • Battery status 	Detect obstacle Detect fire Detect fire Detect battery
MotorControl Pattern	Control robot movement
HumanRobotInterface (HRI) Pattern	Display robot status
InputOutput (IO) Pattern	Switches configure robot
Actuator pattern	For motor
BahaviorBasedControl (BBC) pattern	For subsumption architecture

The POAD methodology suggest the pattern these pattern instances creation is performed at the design phase, but the integration POAD and PECOS Meta model (2005) perform this creation at analysis phase to visualize the mapping process of the use case diagram and the analysis pattern selected. The instances of this phase also capture the main requirements of the AMR problem and model the problem using the analysis pattern.

Based on pattern instances of the selected analysis pattern, the relationship between these instances are identifies to produce the pattern level diagram with specifies the AMR pattern instances and the relationship for this case study is develop as shown in Figure 5.6. The diagram shows the eleven packages involve in the AMR application: DisplayStatus, BatterySensor, Switches, DistanceSensorr, TemperatureSensor, LightSensor, SubsumptionArchitecture, LeftMotorControl, RightMotorControl, RightMotor and LeftMotor. The diagram also show the pattern name which tag by <<>> symbols, the pattern involve in AMR application are HumanRobotInterface (HRI), Sensor, InputOutput (IO), BehaviorBasedControl (BBC), MotorControl and Actuator.

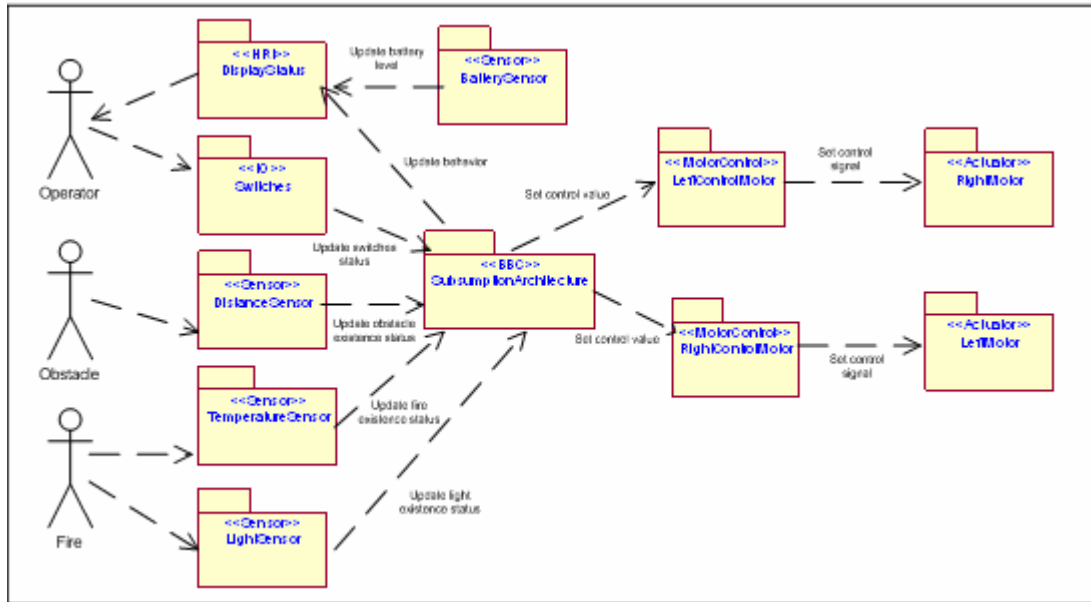


Figure 5.6: Pattern level diagram for AMR

5.3.2 Early Design Phase for AMR

The early design phase used the pattern level diagram that produced by analysis phase, as an input to construct pattern level diagram with interface and pattern level diagram with control interface for the AMR application. The interfaces are the means by which the components connect. The composition of the components using AMR analysis pattern is supported by interfaces defined in the analysis pattern solution. The process start by identify the pattern interface, two types of interface were identified there are functional interface and control interface. The functional interface was specified based on the solution proposed in AMR analysis pattern, while control interface contain of real-time requirement which support the design model. The pattern level diagram with interface for AMR application is shown in Figure 5.7.

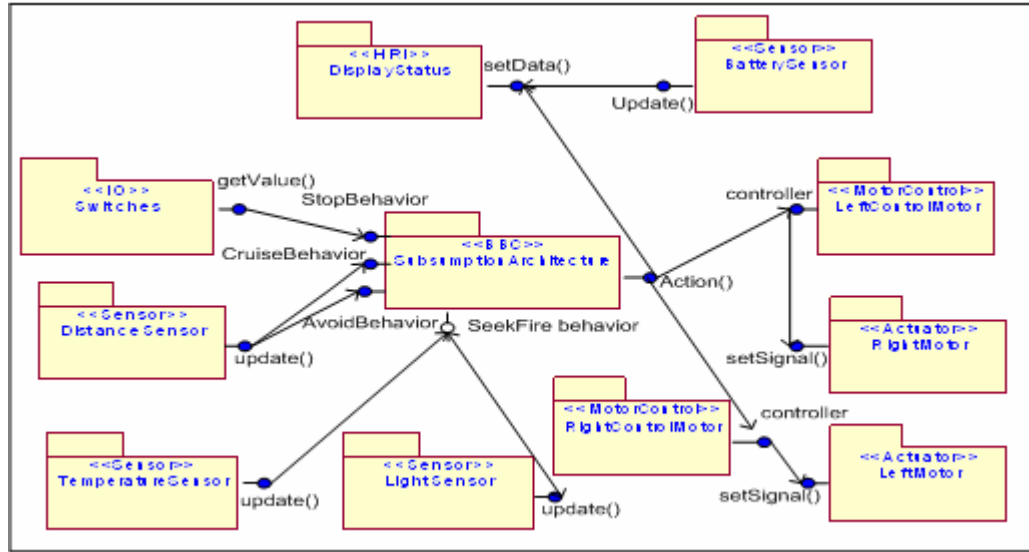


Figure 5.7: Pattern level diagram with interface for AMR

The construction pattern level diagram with interface start by detail out the class diagram at Figure 5.6 into lower level design relationship. The interface from the diagram at Figure 5.7 will be transform into protocol in UML-RT, for example Figure 5.8 shows the interface for Distance Sensor transform into protocol called Sensor. The capsule assumed as a packages in UML-RT, based on Figure 5.8 the distance capsule and avoid capsule can communicate through the sensor protocol as a connector.

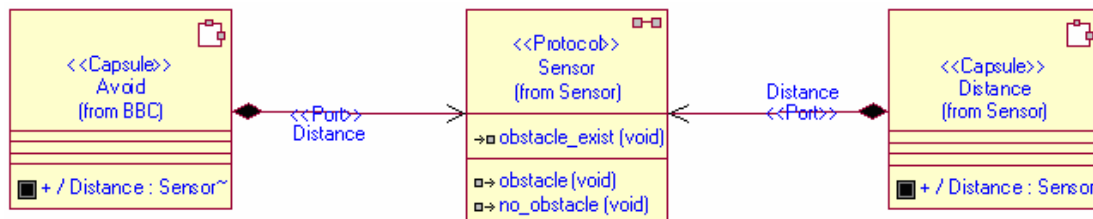


Figure 5.8: Example interface for Distance Sensor

Once the pattern level diagram with interface were obtained, each of the generic analysis pattern in the diagram needs to be renamed, classes in the pattern need to be detailed out according to the specific AMR system and tracing of pattern interfaces to internal classes need to be defines.

The construction pattern level diagram with control interface starts by detailed out the class diagram from Figure 5.7. The Figure 5.9 was obtained by detailing the real-time requirement on the AMR application. The diagram 5.9 show the control interface which was introduced for wiring real-time components to the design pattern. The control interface defines the attributes of real-time requirements of a pattern component and it only necessary in active and event classes.

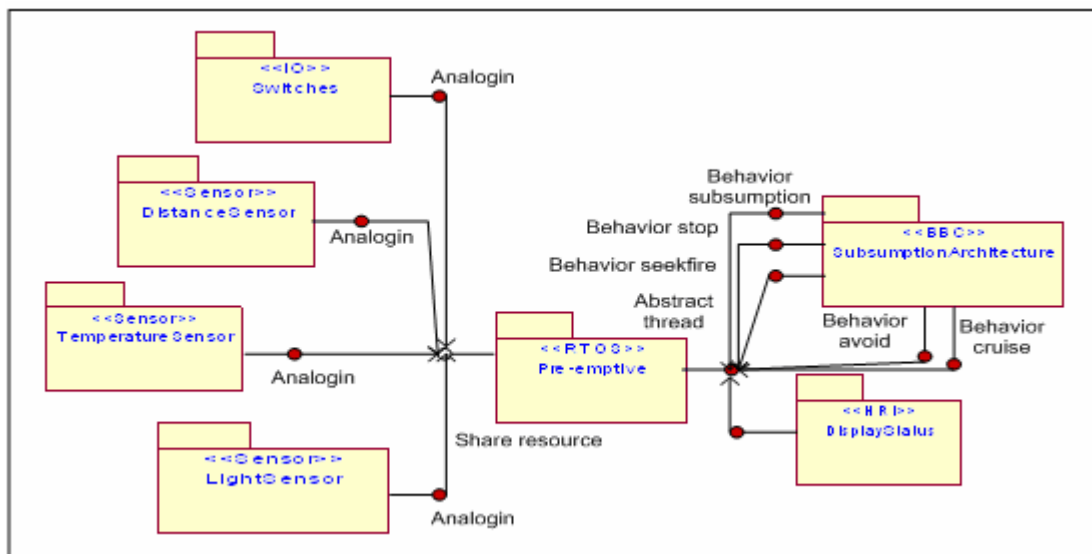




Figure 5.9: Pattern level diagram with control interface for AMR

Once the pattern level diagram with control interface was obtained, the packages in the Figure 5.9 will be detailed out into lower level diagram, while the control interface will be detailed out with the stereotype  in UML-RT, but since

UML-RT not enough notations to design the symbols, the control interface which contains the timing properties are only tag with  symbol. The Figure 5.10 has shown those processes for Figure 5.9.

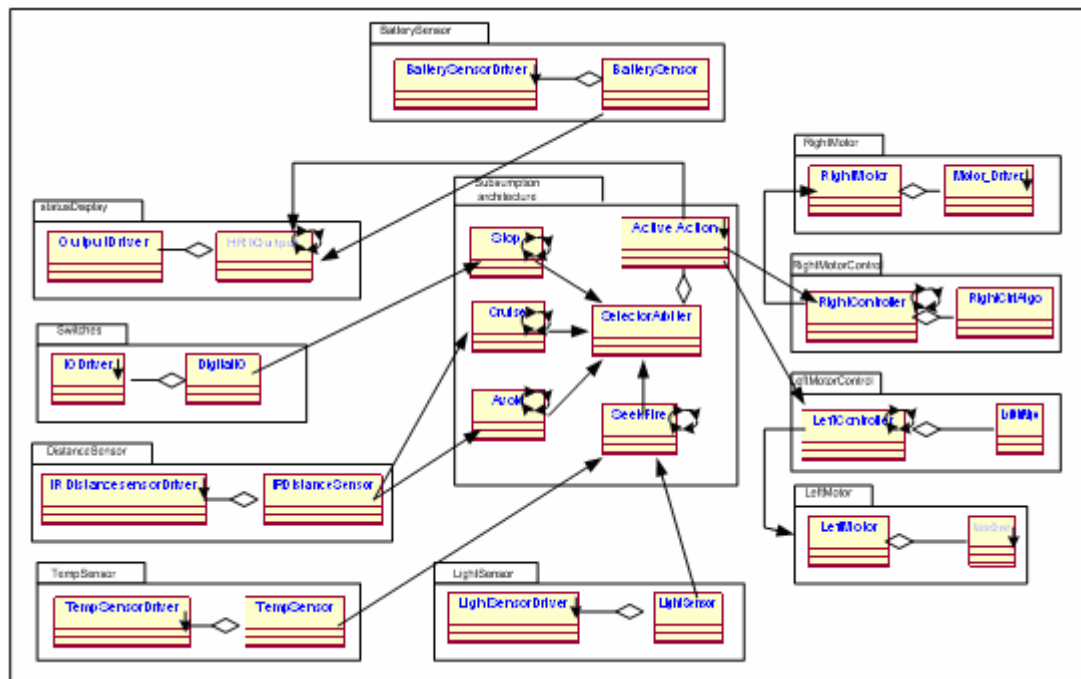


Figure 5.10: Detail Control Interface for Pattern level diagram.

In order to complete the analysis model, behavior model of each component and the integrated components need to be defined. There are various behavioral diagrams can be used to design the behavior model, such as activity diagrams, sequence diagrams and state machine diagram. Based on that, the behavior model of AMR is constructed using UML-RT such as in Figure 5.11. The Figure 5.11 was obtained by transforming the Figure 5.7 to UML-RT capsules, ports and connectors. The Figure shows the relationship between the capsule which is treated as a components in AMR application. Each of components communicates through the port as an interface and communicates through the connector as a protocol at UML-

RT. The diagram at Figure 5.9 cannot transform into UML-RT because UML-RT tools cannot represent the real-time behavior as shown in Figure 5.10.

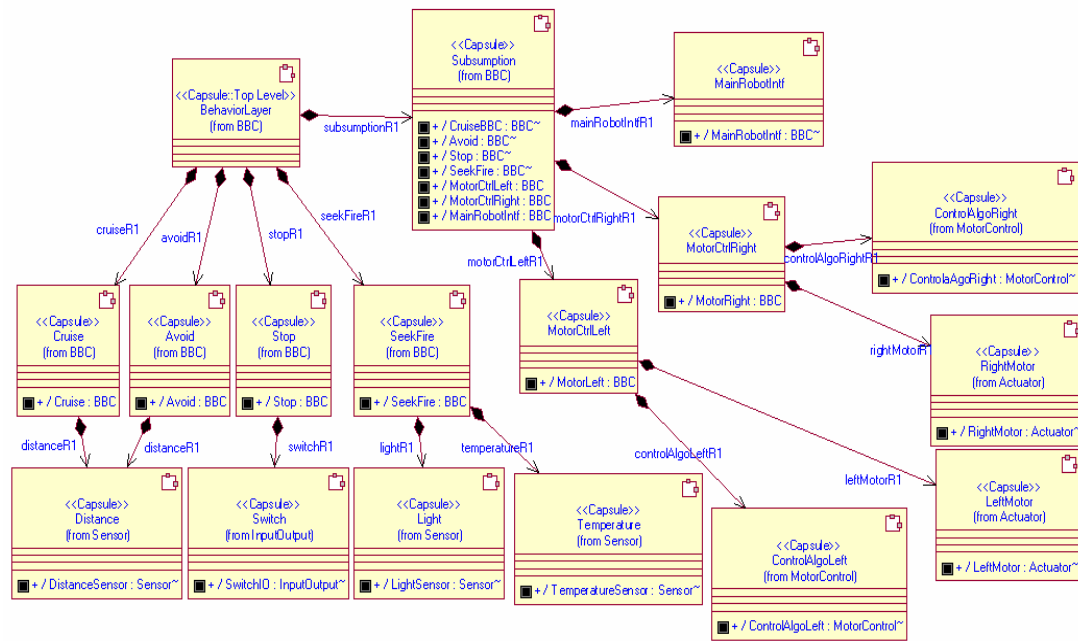


Figure 5.11: The Behavior of AMR Design using UML-RT

5.3.2.1 Simulation of Avoid Behavior

In order to show the simulation for AMR application, the Avoid behavior was selected. The simulation is applied in order to how UML-RT can support the software process for integrated POAD and PECOS. Figure 5.12 shows the relationship in Avoid behavior. The simulation process can be done by defining the protocol between the avoid behavior component, for example, Sensor is declared as a protocol between Avoid component and distance component.

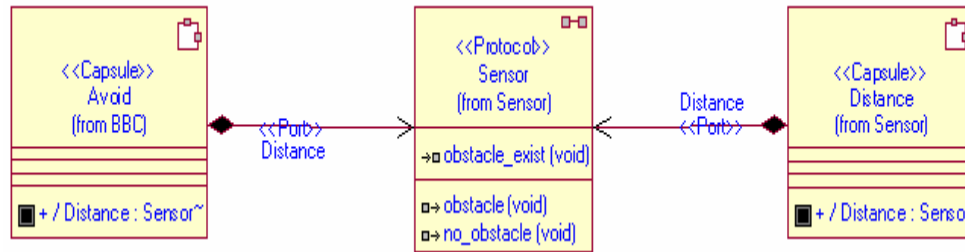


Figure 5.12: Sensor Protocol between Avoid and Distance

Figure 5.13 shows the structure diagram for Avoid behavior, the Figure illustrate the Avoid as an end port and Distance as end port. The Figure 5.14 shows the structure diagram for Distance, which contains Distance as a start port.

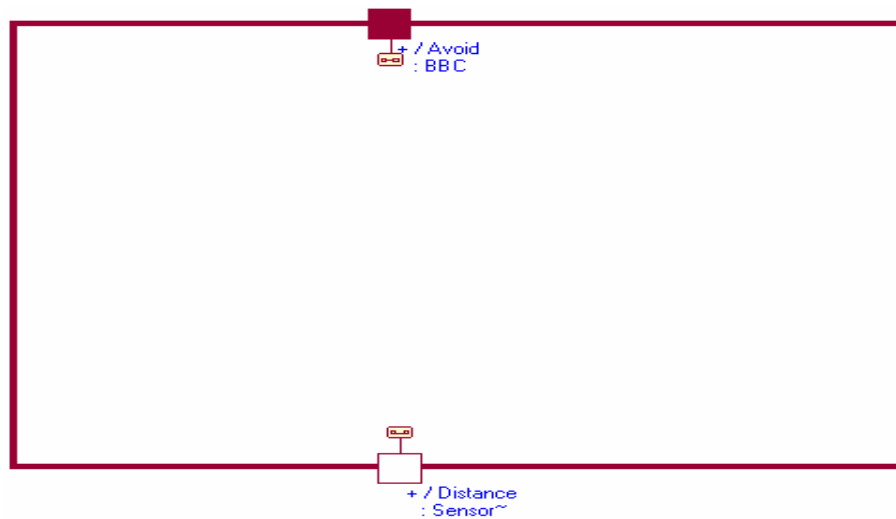


Figure 5.13: Structure diagram for avoid

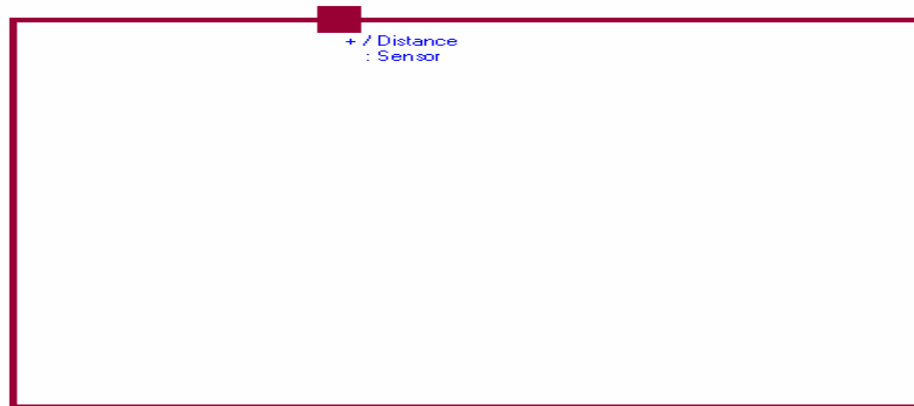


Figure 5.14: Structure diagram for Distance

In this project UML-RT is used to validate the functional by doing the simulation. For example based on state diagram for avoid behavior we doing the simulation by enter input called Distance, such show in Figure 5.15. After send the signal which called existObstacle into state diagram, the state moving from no_obstacle state into exist_obstacle state. This activity was shows at Figure 5.16, Figure 5.17 and Figure 5.18. Figure 5.16 show the state diagram before doing the simulation, while Figure 5.17 show the state diagram when receive the input and 5.18 show the diagram after receive input. The moving from no_obstacle into exist_obstacle is show during the simulation.

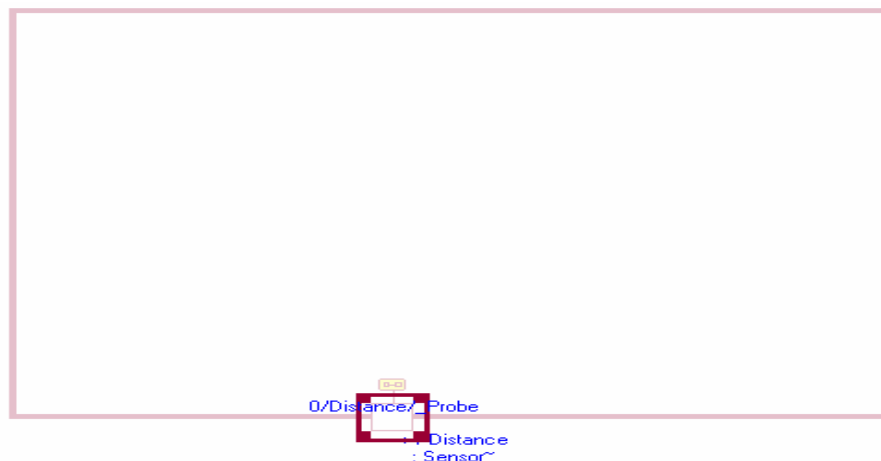


Figure 5.15: Input port after receive input

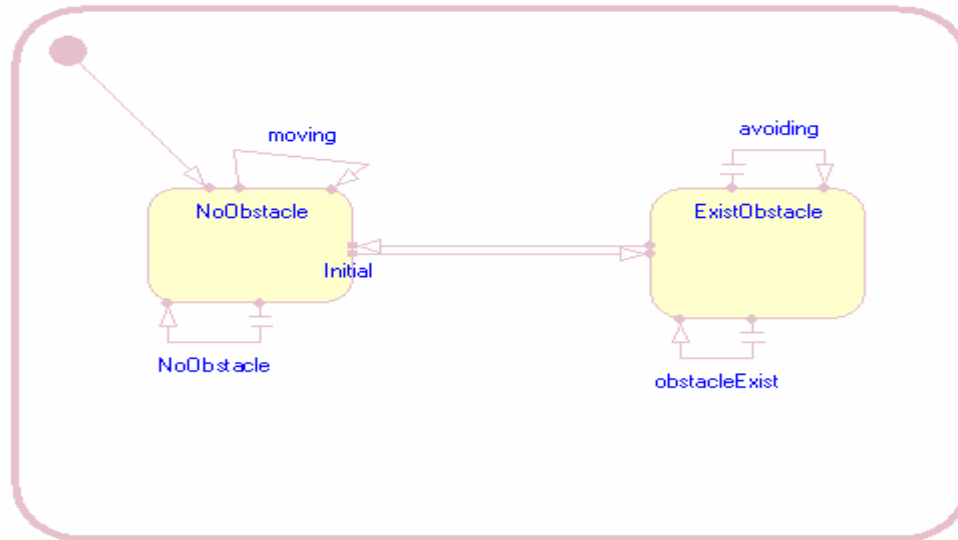


Figure 5.16: State diagram for avoid before receive input

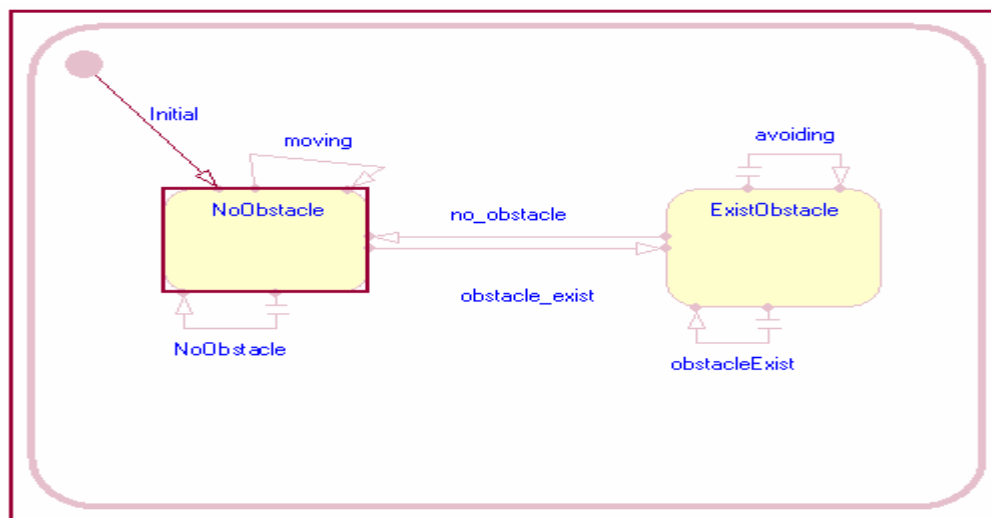


Figure 5.17: State diagram Avoid with input NoObstacle

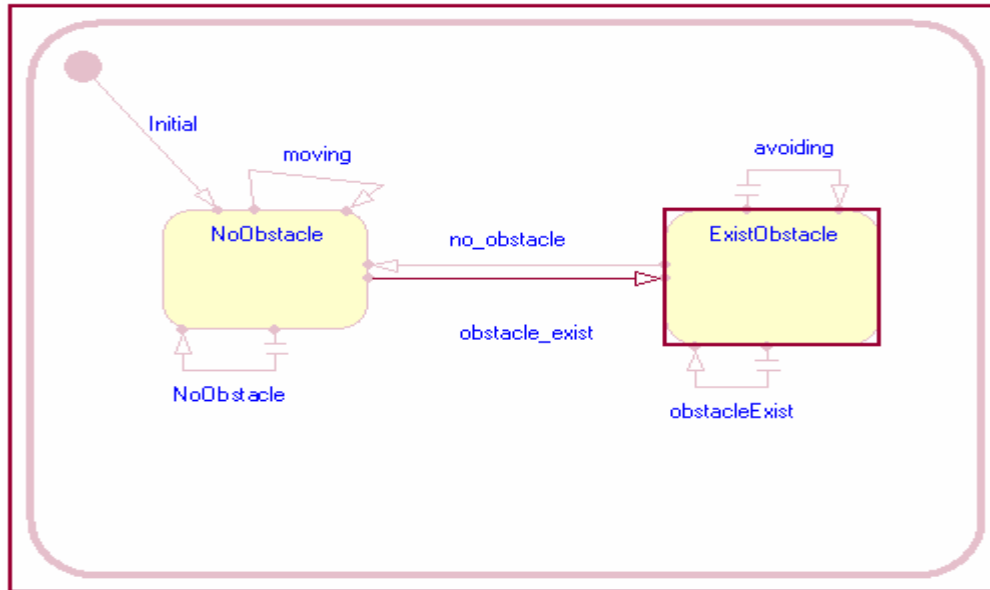


Figure 5.18: State diagram Avoid with input ExistObstacle

5.3.3 Detailed Design Phase for AMR

The objective of detailed design phase is to create a detailed model of the software using the information that provide by early design phase. The detail design model for AMR application is created based on reveal structure of the pattern component groups such as in Figure 5.10 at above. The structure class diagram and real-time requirement will be transformed to PECOS integration diagram to develop a detail component-level design. The product of the transformation process is a simple block diagram, as shown in Figure 5.19.

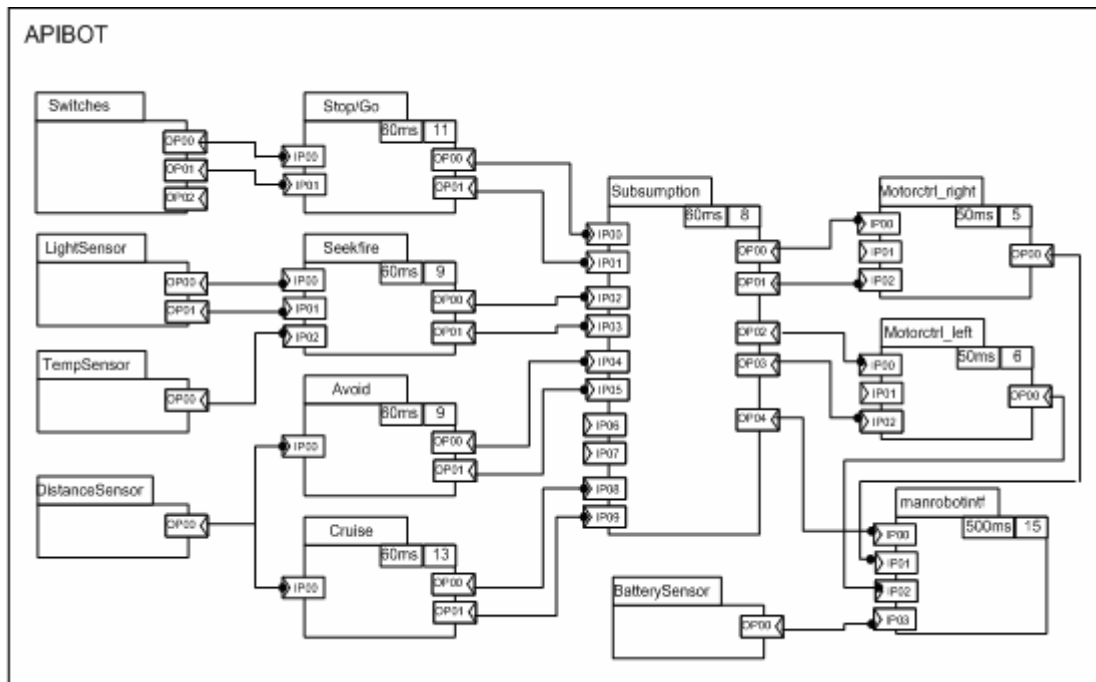


Figure 5.19: The APIBOT composition based in design components artifact

Figure 5.19 was obtained by set the timing for the component, for example the LightSensor component are supported by subsumption component through the SeekFire, such as Figure 5.10. The detailed design phase cannot be design using UML-RT because the UML-RT tools not enough notations to design those models. In this project the Figure 5.16 is design using different drawing tools. At detailed design phase, the output and input port are set, such as Figure 5.19.

5.4 Discussion on Process Model using UML-RT

This chapter is about a process model using UML-RT for a AMR case study. A simulation was used in order to show that UML-RT can support the process integration between POAD and PECOS. In applying it in the AMR case study, it

showed that UML-RT notations is not enough to support the integrated approach, which the detail design for AMR application cannot be designed using UML-RT. Moreover, an UML-RT notation is also not enough to support the construction of some diagram in early design phase. For example UML-RT cannot support the design of packages and interfaces during design pattern level diagram with interface and pattern level diagram with control interface. However, during the early design phase, UML-RT is a suitable approach that can support the design of behavior model for integrated POAD and PECOS. For example the avoid behavior can design using UML-RT.

CHAPTER 6

CONCLUSION

6.1 Summary

The defining process into formal form is important to enable and support the development CASE tools for ERT software and promote software reused in ERT systems without sacrificing the non-functional requirement such as timeless, predictability and constrained resources. This project focuses on identifying and defining the process for integrated POAD and PECOS Meta model. In order to implement this project, the process can be identify and define based on existing integrated Meta model, which was done by Jawawi (2005).

Based on combination POAD and PECOS Meta model, the process divide into three phases, there are analysis phase, early design phase and detail design phase. The analysis and early design phase was develop using POAD methodology, while detail design phase used PECOS Meta model. In order to design with PECOS model, the transform from POAD and PECOS is need. The transform process can be done by mapping the interface and real time behavior between these two models. In this

project, the processes are defined using Software Process Engineering Metamodel (SPEM) icons. The process definition is based on phase. Every step at the phases will be detailed out and figured out using related icon at SPEM. Based on defining process using SPEM, the process is demonstrated by applying the case study. The Autonomous Mobile Robot is the case study that is chosen in this project.

6.2 Project Architecture and Contribute

This project is only limited to software process for integrated POAD and PECOS. The process was obtained from the existing integrated Meta model POAD and PECOS (2005). The architecture of the project concerns about the analysis phase and early design of POAD and Detailed Design Phase using PECOS model. This project only focuses on analysis, design and detailed design phase and the implementation phase is not covered in this project. The process starts by study the POAD methodology and PECOS model, and followed by the study of integrated POAD and PECOS Meta model. Based on the integration showed, every step and the activity was figured out, identified and defined into formal form using SPEM icons which was discussed in Chapter 4. The main contribution in this project is software process for integrated POAD and PECOS Meta model (2005).

Based on the process model which was defined using SPEM, the process model was implementing using UML-RT. The implementation process model using UML-RT can be done by applying the Autonomous Mobile Robot (AMR) case study. As a result UML-RT can support the design of behavior model for AMR. However, UML-RT notations are not enough to support the construction interfaces and packages during early design phase and design the diagram at detailed design phase.

6.3 Future Work

In this project, the UML-RT is used to design the software process for integrated POAD and PECOS. The UML-RT only can be used as tools to design the behavior of AMR application at early design stage. The future work on this project is to develop tools for integrated POAD and PECOS. Since UML-RT cannot support the real-time properties, another future work is demonstrate the process using different tools, such as UML-RT Rhapsody and extends POAD and PECOS method, to enable systematic transformation of functional behavior.

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.










ID	Task Name	Duration	Start	Finish	May 4, '08	May 11, '08	May 18, '08	May 25, '08	Jun 1, '08	Jun 8, '08	Jun 15, '08	Jun 22, '08
					S	S	S	S	S	S	S	S
1	project 1	38 days?	Mon 5/5/08	Wed 6/25/08	[Gantt bar for project 1]							
2	Analysis problems and conduct literature review	10 days?	Mon 5/5/08	Fri 5/16/08	[Gantt bar for task 2]							
3	Propose project	1 day?	Tue 5/20/08	Tue 5/20/08	[Gantt bar for task 3]							
4	Project proposal	1 day?	Tue 5/27/08	Tue 5/27/08	[Gantt bar for task 4]							
5	Documentation	11 days?	Wed 5/28/08	Wed 6/11/08	[Gantt bar for task 5]							
6	initial finding	1 day?	Mon 6/16/08	Mon 6/16/08	[Gantt bar for task 6]							
7	submission report	1 day?	Tue 6/17/08	Tue 6/17/08	[Gantt bar for task 7]							
8	slide preparation	3 days?	Mon 6/23/08	Wed 6/25/08	[Gantt bar for task 8]							
9	project 1 presentation	1 day?	Fri 6/20/08	Fri 6/20/08	[Gantt bar for task 9]							
10	project 1 report correction	3 days?	Mon 6/23/08	Wed 6/25/08	[Gantt bar for task 10]							
11	Project 1 final report submission	1 day?	Wed 6/25/08	Wed 6/25/08	[Gantt bar for task 11]							

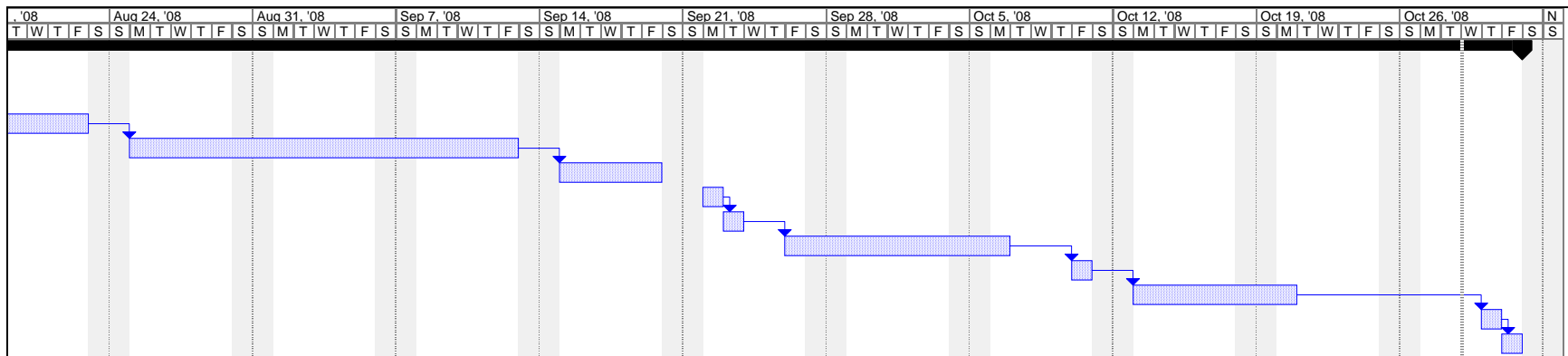
Project: Project1
Date: Wed 10/29/08

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	





ID	Task Name	Duration	Start	Finish	Jul 6, '08							Jul 13, '08							Jul 20, '08							Jul 27, '08							Aug 3, '08							Aug 10, '08							Aug 17, '08		
					S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M								
1	Project 2	85 days?	Mon 7/7/08	Fri 10/31/08	[Gantt bar for Project 2]																																												
2	Review chapter 1, 2, 3	10 days?	Mon 7/7/08	Fri 7/18/08	[Gantt bar for Review chapter 1, 2, 3]																																												
3	Define the process using SPEM	15 days?	Mon 7/21/08	Fri 8/8/08	[Gantt bar for Define the process using SPEM]																																												
4	Documentation for chapter 4	10 days?	Mon 8/11/08	Fri 8/22/08	[Gantt bar for Documentation for chapter 4]																																												
5	Design the process using case st	15 days?	Mon 8/25/08	Fri 9/12/08	[Gantt bar for Design the process using case st]																																												
6	Prepare for chapter 5 & 6	5 days?	Mon 9/15/08	Fri 9/19/08	[Gantt bar for Prepare for chapter 5 & 6]																																												
7	Slide preparation	1 day?	Mon 9/22/08	Mon 9/22/08	[Gantt bar for Slide preparation]																																												
8	Submission report	1 day?	Tue 9/23/08	Tue 9/23/08	[Gantt bar for Submission report]																																												
9	Holiday for Hari Raya	7 days?	Fri 9/26/08	Mon 10/6/08	[Gantt bar for Holiday for Hari Raya]																																												
10	Project 2 presentation	1 day?	Fri 10/10/08	Fri 10/10/08	[Gantt bar for Project 2 presentation]																																												
11	Project 2 report correction	6 days?	Mon 10/13/08	Mon 10/20/08	[Gantt bar for Project 2 report correction]																																												
12	Prepare for thesis report	1 day?	Thu 10/30/08	Thu 10/30/08	[Gantt bar for Prepare for thesis report]																																												
13	Thesis report submission	1 day?	Fri 10/31/08	Fri 10/31/08	[Gantt bar for Thesis report submission]																																												

Project: Projectii
Date: Wed 10/29/08

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	



Project: Projectii
Date: Wed 10/29/08

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	