A FRAMEWORK FOR COMPONENT-BASED REUSE FOR AUTONOMOUS MOBILE ROBOT SOFTWARE

DAYANG NORHAYATI ABANG JAWAWI

UNIVERSITI TEKNOLOGI MALAYSIA
A FRAMEWORK FOR COMPONENT-BASED REUSE FOR AUTONOMOUS MOBILE ROBOT SOFTWARE

DAYANG NORHAYATI ABANG JAWAWI

A thesis submitted in fulfilment of the requirements for the award of the degree of Doctor of Philosophy

Faculty of Computer Science and Information System
Universiti Teknologi Malaysia

DECEMBER 2006
Specially dedicated to my family
ACKNOWLEDGEMENTS

I wish to express my thanks and gratitude to my supervisor, Professor Dr. Safaai Deris, for his guidance during course of this work.

I would like to acknowledge the Public Service Department of Malaysia and Universiti Teknologi Malaysia for their financial support.

I would like to thanks the Mobile Robotics Research Group in Universiti Teknologi Malaysia for their helps and support.

Finally, my special thanks to my parents for their love and care, my husband, Rosbi Mamat, for his patience and love, and my kids, Munirah, Adibah, Ismael and Muhammad for cheering me up at those difficult time.
ABSTRACT

Applying software reuse to Embedded Real-Time (ERT) systems poses significant challenges to industrial software processes due to the resource-constrained and real-time requirements of the systems. Autonomous Mobile Robot (AMR) system is a class of ERT systems, hence, inherits the challenge of applying software reuse in general ERT systems. Furthermore, software reuse in AMR systems is challenged by the diversities in terms of robot physical size and shape, environmental interaction and implementation platform. Thus, it is foresee that component-based reuse will be the suitable way to promote software reuse in AMR systems with consideration to three AMR general requirements to be self-contained, platform-independence and real-time predictable. In this thesis, a framework for component-based reuse of AMR software has been developed to enable a systematic reuse through component-based software engineering. The aim of the framework is to outline the strategies for software reuse in software development of AMR applications. The developed framework consists of four main elements: AMR component-based analysis patterns, a modified component model, a component-based timing analysis approach, and a component-oriented programming framework. The results of implementing the framework in developing software for real AMR show that the strategies and processes proposed in the framework can fulfill the three AMR general requirements. To quantify the effectiveness of the reuse approach in the developed framework, the component reusability and the amount of reuse were measured using software metrics. The measurement results show high component reusability on those interested components, and up to 74% of reuse rate was achieved on real AMR tested. The implementation results and software reuse measurements indicate that the developed framework promotes systematic reuse and reuse qualities.
ABSTRAK

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Background of the Problem</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Some Challenges in Embedded Real-Time Software</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Challenge in Autonomous Mobile Robot Software</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Statement of the Problem</td>
<td>9</td>
</tr>
<tr>
<td>1.4</td>
<td>Objectives</td>
<td>10</td>
</tr>
<tr>
<td>1.5</td>
<td>Scope of the Study</td>
<td>11</td>
</tr>
<tr>
<td>1.6</td>
<td>Thesis Outline</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>SOFTWARE REUSE AND COMPONENT-BASED SOFTWARE ENGINEERING FOR AUTONOMOUS MOBILE ROBOT SYSTEMS</td>
<td>13</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Software Reuse Strategies</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Substance of Reuse and Product of Reuse</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Technique of Reuse</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Scope of Reuse</td>
<td>17</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Mode of Reuse and Intention of Reuse</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Software Reuse in Robotic Applications</td>
<td>18</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Framework</td>
<td>19</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Architecture</td>
<td>22</td>
</tr>
</tbody>
</table>
2.3.3 Software Patterns
2.3.4 Component-Oriented Programming
2.3.5 Component Reuse in Commercial Robotics Product
2.3.6 Summary of Robotic Reuse Strategies
2.4 Component-based Software Engineering
2.4.1 Component Technology
2.4.2 Component Model for Resource-Constrained Embedded Real-Time Systems
2.5 Software Patterns in Component Reuse
2.5.1 Pattern-Oriented Development Methodology
2.6 Software Architecture in Component Reuse
2.7 Predictable Real-Time Software in Component-Based Software Development
2.7.1 Real-Time Scheduling Theory as Analytical Model
2.8 Discussion
2.9 Summary

3 RESEARCH METHODOLOGY
3.1 Introduction
3.2 Research Design
3.3 Research Processes
3.3.1 Determine Reusable Components
3.3.2 Evaluate and Enhance Embedded Real-Time Component Model
3.3.3 Determine Timing Analysis Approach
3.3.4 Develop Framework for Component-Based Reuse
3.4 The Software Engineering Research
3.5 Autonomous Mobile Robot Case Studies
3.5.1 IMR71848 Robot
4 COMPONENT-BASED ANALYSIS PATTERN FOR AUTONOMOUS MOBILE ROBOT SYSTEMS

4.1 Introduction

4.2 Analysis Pattern for Autonomous Mobile Robot Software
   4.2.1 Analysis Patterns Mining Process
   4.2.2 Defining Analysis Pattern at Reactive Layer
   4.2.3 Defining Component-Based Analysis Patterns

4.3 Deployment of the Analysis Patterns in Component-Based Development
   4.3.1 The Mobile Robot Software Analysis
   4.3.2 The Mobile Robot Software Early Design

4.4 Application of the Analysis Patterns for Pattern-Based Reverse Engineering
   4.4.1 Reverse to Class Phase
   4.4.2 Reverse to Pattern Phase
   4.4.3 Reverse to Pattern Interface Phase

4.5 Discussion

4.5 Summary

5 COMPONENT-MODEL SELECTION AND A COMPONENT-ORIENTED PROGRAMMING FRAMEWORK

5.1 Introduction

5.2 Methodology for Evaluating the PECOS and ReFlex Component Models

5.3 Evaluation of ReFlex and PECOS Component Models
   5.3.1 Component Model
AN APPROACH FOR PREDICTABLE REAL-TIME PERFORMANCE OF AUTONOMOUS MOBILE ROBOT SOFTWARE

6.1 Introduction 118

6.2 Experiences from a Predictable Assembly of Component 119

6.3 Temporal Modeling of Autonomous Mobile Robot Software 121

6.4 An Approach for Prediction of Robot Behaviors from Components Assembly 128

6.4.1 Utilization Feasibility Analysis 129

6.4.2 Exact Feasibility Analysis 130
Experimental Results 131
6.5.1 Utilization Feasibility Analysis 132
6.5.2 Exact Feasibility Analysis 134
6.6 Discussion 136
6.7 Summary 137

A FRAMEWORK FOR COMPONENT-BASED REUSE 138
7.1 Introduction 138
7.2 A Framework for Software Reuse 139
7.3 Transforming POAD Pattern-oriented Models to PECOS Component-based Models 141
7.3.1 The Enhanced POAD Construct 141
7.3.2 The Modified PECOS Component Model 145
7.3.3 Mapping POAD to PECOS Model 147
7.4 Application of the Framework 149
7.4.1 Analysis Phase 150
7.4.2 Early Design Phase 153
7.4.3 Detailed Design Phase 157
7.4.4 Implementation Phase 160
7.5 Discussion 161
7.6 Summary 162

MEASUREMENT OF SOFTWARE REUSE AND RESULTS ON THE DEVELOPED FRAMEWORK 163
8.1 Introduction 163
8.2 Measurement of Software Reuse through Metrics 163
8.3 Measuring Component Reusability 165
8.4 Measuring Component Amount of Reuse 169
8.4.1 Platform Independent Components 172
8.4.2 Platform-Independence Components with Platform-Dependence Modules 174
8.5 Discussion 175
8.6 Summary

9 CONCLUSION
9.1 Summary
9.2 Research Conclusion
9.3 Research Contribution
9.4 Future Works

REFERENCES

APPENDICES A-C
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Summary on two previous evaluations on ERT component model</td>
<td>35</td>
</tr>
<tr>
<td>2.2</td>
<td>PECT technology for different prediction activities</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>Summary of the research process</td>
<td>54</td>
</tr>
<tr>
<td>3.2</td>
<td>Research strategies for this research in Shaw’s model</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>Summary of purpose for each pattern</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Number of components from the pattern used in both software</td>
<td>85</td>
</tr>
<tr>
<td>5.1</td>
<td>Evaluation summary</td>
<td>100</td>
</tr>
<tr>
<td>5.2</td>
<td>The result of PECOS component mapping for the AMR system</td>
<td>104</td>
</tr>
<tr>
<td>5.4</td>
<td>Performance and price of the short listed RTK.</td>
<td>108</td>
</tr>
<tr>
<td>6.1</td>
<td>The AMR threads and their timing properties</td>
<td>126</td>
</tr>
<tr>
<td>6.2</td>
<td>The mapping between UML-RT and PECOS model elements</td>
<td>127</td>
</tr>
<tr>
<td>6.3</td>
<td>Various setting for $T_a$ and $T_b$</td>
<td>132</td>
</tr>
<tr>
<td>6.4</td>
<td>Intersection point of $T_a$ group when $u_1$ and $u_2$</td>
<td>133</td>
</tr>
<tr>
<td>6.5</td>
<td>MobileRobot’s temporal properties</td>
<td>134</td>
</tr>
<tr>
<td>6.6</td>
<td>The timing results</td>
<td>135</td>
</tr>
<tr>
<td>7.1</td>
<td>General mapping of POAD and PECOS model elements</td>
<td>147</td>
</tr>
<tr>
<td>7.2</td>
<td>The APIBOT threads and their timing properties</td>
<td>157</td>
</tr>
<tr>
<td>7.3</td>
<td>The timing properties for APIBOT components</td>
<td>159</td>
</tr>
</tbody>
</table>
8.1 Washizaki’s metrics applied on the AMR component-based analysis pattern 168
8.2 The AMR systems processor platform and size 170
8.3 The AMR systems sensors and actuators 170
8.4 APIBOT LOC by components 171
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Examples of embedded systems</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>VRF limiting the effects of diversity</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>Component-based framework for AMR</td>
<td>20</td>
</tr>
<tr>
<td>2.3</td>
<td>CLARAty architecture</td>
<td>23</td>
</tr>
<tr>
<td>2.4</td>
<td>LAAS architecture</td>
<td>24</td>
</tr>
<tr>
<td>2.5</td>
<td>The STESCA pattern overview</td>
<td>26</td>
</tr>
<tr>
<td>2.6</td>
<td>Software architecture of the pattern-oriented framework</td>
<td>26</td>
</tr>
<tr>
<td>2.7</td>
<td>PBO component model</td>
<td>27</td>
</tr>
<tr>
<td>2.8</td>
<td>Two main activities in CBSE</td>
<td>31</td>
</tr>
<tr>
<td>2.9</td>
<td>Relations between basic concepts of component technology</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Overview of the research processes in developing the framework</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>The IMR71848 robot</td>
<td>58</td>
</tr>
<tr>
<td>3.3</td>
<td>Block Diagram of the IMR71848 robot controller</td>
<td>59</td>
</tr>
<tr>
<td>3.4</td>
<td>The APIBOT robot</td>
<td>60</td>
</tr>
<tr>
<td>3.5</td>
<td>Block Diagram of the APIBOT</td>
<td>60</td>
</tr>
<tr>
<td>4.1</td>
<td>Analysis patterns in the software development process</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Behavior-based robotic control</td>
<td>67</td>
</tr>
<tr>
<td>4.3</td>
<td>Analysis pattern for reactive layer</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>Behavior-based control pattern catalogue</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>The AMR use case diagram</td>
<td>73</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Analyzed subsystems and AMR patterns matching</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>Pattern-level diagram for the AMR software</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>Pattern-level with interface diagram for AMR intelligence behavior</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>Detailed internal classes representation of the AMR for intelligence behavior</td>
<td></td>
</tr>
<tr>
<td>4.10</td>
<td>Pattern-level with control interface diagram for intelligence behavior</td>
<td></td>
</tr>
<tr>
<td>4.11</td>
<td>Fire Marshal Bill class diagram</td>
<td></td>
</tr>
<tr>
<td>4.12</td>
<td>The Arbitration and Motor packages</td>
<td></td>
</tr>
<tr>
<td>4.13</td>
<td>Pattern Level diagram for Fire Marshal Bill mobile robot</td>
<td></td>
</tr>
<tr>
<td>4.14</td>
<td>Pattern-Level with Functional Interface diagram for the AMR</td>
<td></td>
</tr>
<tr>
<td>4.15</td>
<td>Pattern-Level with Control interface diagram for the intelligent mobile robot</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Behavior Layers in AMR software</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Components assembly for the AMR case study</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>The ReFlex component model</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>The PECOS component model</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Definitions of ports and interfaces using ART-ML for Pi component</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Pi task and Pi component using ART-ML in ReFlex</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Pi component definition using PECOS diagrammatical expression</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>Pi component definition using PECOS CoCo language</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>Component instances and components’ connection using ART-ML</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>Example of components assembled for motor speed control subsystem in PECOS</td>
<td></td>
</tr>
</tbody>
</table>
5.11 MSC component definition using PECOS CoCo language
5.12 Petri net of motor speed control subsystem
5.13 Components integration using PECOS model
5.14 Hierarchical view of the AMR components
5.15 Components integration of an AMR application case study
5.16 Hardware and RTOS abstraction layers of the proposed COP framework
5.17 A PID component documented in block form
5.18 MotorControl composite component
5.19 Code template for MotorControl composite component
5.20 The code skeleton for MobileRobot component task execution
6.1 The predictable assembly of components approach based on PECT
6.2 The AMR behavior layer architecture
6.3 Relationship between the capsules in the AMR system
6.4 UML-RT structure model of the AMR system
6.5 Sequence diagram representing message sequences between the capsules
6.6 Protocol used to connect ports between Switches and Stop capsules
6.7 Components composition of the AMR using PECOS model
6.8 Solution for number of maximum $N_b$ for the case-study AMR in terms of utilization $U$ and 100% for nine various setting of $T_a$ and $T_b$
6.9 MobileRobot’s task schedule
7.1 The Framework overview
7.2 Overview of POAD Metamodel
7.3 An application engineering activities based on the framework  
7.4 The APIBOT use case diagram  
7.5 Pattern-level diagram for the APIBOT software  
7.6 Pattern-level diagram with interface for the APIBOT software  
7.7 The APIBOT classes in pattern component group  
7.8 Pattern-level diagram with control interface for the APIBOT software  
7.9 Relationship between capsules  
7.10 Structure model of the APIBOT system  
7.11 State machine to define Avoid behavior of the APIBOT system  
7.12 APIBOT component integration using PECOS model  
7.13 The APIBOT composition based in design components artifact  
8.1 Actuator pattern component  
8.2 Reuse on APIBOT system  
8.3 Components and modules in APIBOT software composition  
8.4 APIBOT platform-independence software  
8.5 Percentage of components in each APIBOT component groups  
8.6 APIBOT platform-independence with platform dependent software  
8.7 Percentage reuse of components in each APIBOT component group with platform-dependence component
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERT</td>
<td>Embedded Real-Time</td>
</tr>
<tr>
<td>AMR</td>
<td>Autonomous Mobile Robot</td>
</tr>
<tr>
<td>ART-ML</td>
<td>Architecture and Real-Time behavior Modeling Language</td>
</tr>
<tr>
<td>BBC</td>
<td>Behavior-Based Control</td>
</tr>
<tr>
<td>CBD</td>
<td>Component-Based software Development</td>
</tr>
<tr>
<td>CBSE</td>
<td>Component-Based Software Engineering</td>
</tr>
<tr>
<td>CCM</td>
<td>CORBA Component Model</td>
</tr>
<tr>
<td>CLARAty</td>
<td>Coupled Layered Architecture for Robotic Autonomy</td>
</tr>
<tr>
<td>CoCo</td>
<td>Component Composition</td>
</tr>
<tr>
<td>COM</td>
<td>Component Object Model</td>
</tr>
<tr>
<td>CoolBOT</td>
<td>Component-Oriented Programming Framework for Robotics</td>
</tr>
<tr>
<td>COP</td>
<td>Component-Oriented programming</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial off-the-shelf</td>
</tr>
<tr>
<td>DC</td>
<td>Direct-Current</td>
</tr>
<tr>
<td>DCOM</td>
<td>Distributed Component Object Model</td>
</tr>
<tr>
<td>DX</td>
<td>Data Exchange</td>
</tr>
<tr>
<td>EDF</td>
<td>Earliest Deadline First</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise JavaBeans</td>
</tr>
<tr>
<td>FBD</td>
<td>Function Block Diagram</td>
</tr>
<tr>
<td>HAL</td>
<td>Hardware Abstraction Layer</td>
</tr>
<tr>
<td>HLRC</td>
<td>High Level Robot Control</td>
</tr>
<tr>
<td>HRI</td>
<td>Human-Robot Interface</td>
</tr>
<tr>
<td>IDL</td>
<td>Interface Definition Language</td>
</tr>
<tr>
<td>IR</td>
<td>Infra-Red</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
</tbody>
</table>
LCD - Liquid Crystal Display
LOC - Line Of Code
NiMH - Nickel-Metal Hydride
OO - Object-Oriented
OPEN-R - open architecture
OROCOS - Robot Control Software
OSCAR - Operating System for the Control of Autonomous Robots
PBO - Port-based Object
PD - Proportional-Derivative
PDMA - Pattern-Driven Modeling and Analysis
PECOS - PErvasive COmponent Systems
PECT - Predictable Enable Component Technology
PID - Proportional-Integral-Derivative
POAD - Pattern-Oriented Analysis and Design
PWM - Pulse-Width Modulation
QoS - Quality of Services
RCC - Rate of Component Customizability
RCO - Rate of Component Observability
ReFlex - Flexible real-time component model
rHAL - robotic Hardware Abstraction Layer
RMA - Rate Monotonic Algorithm
ROPES - Rapid Object-Oriented processing for Embedded Systems
RTE - RunTime Environment
RTEMS - Real Time Executive for Multiprocessor Systems
RTK - Real-Time Kernel
RTOS - Real-Time Operating System
RWI - Real World Interface
SEI - Software Engineering Institute
STESCA - Strategic-Tactical-Execution Software Control Architecture
UML - Unified Modeling Language
UML-RT - Unified Modeling Language for Real-Time
UTM - Universiti Teknologi Malaysia
VLSI - Very Large-Scale Integrated
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRF</td>
<td>Virtual Robot Framework</td>
</tr>
<tr>
<td>WCET</td>
<td>Worst-Case Execution Times</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
# LIST OF APPENDICES

<table>
<thead>
<tr>
<th>APPENDIX</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Autonomous Mobile Robot Component-Based Analysis Pattern</td>
<td>197</td>
</tr>
<tr>
<td>B</td>
<td>Autonomous Mobile Robot Design Components</td>
<td>202</td>
</tr>
<tr>
<td>C</td>
<td>Paper Published from This Thesis Work</td>
<td>206</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Overview

The trend of software system development is changing from traditional software development approach, which focuses on building software systems from scratch to extension and integration with preexisting systems software development approach (Finkelstein and Kramer, 2000). This is due to the rapid changes in modern software requirements, which make the software becoming large, complex and highly unmaintainable. The increment in the scale and the complexity of software may lead to problems such as failure of software projects to meet their deadline, budget, quality requirements and the continual increase of the costs associated with software maintenance.

The same phenomenon is faced by Embedded Real-Time (ERT) systems due to the need for more sophisticated products. The machines and equipment controlled by ERT systems have becoming more sophisticated and intelligent, often incorporating many functions in one product. As ERT products increasingly adopt digital technology, the use of advanced microprocessors enable more processing to be implemented in software, making the development of software for ERT systems more important. Also, the shift from full hardware implementation of ERT systems to mixed software and hardware implementation makes the software developer jobs more challenging. A common characteristic of all ERT systems now is increasing demand on software. For example, currently the software development costs for
industrial robots is about 75% of total costs, while in car industry is about 30% of total costs (Crnkovic, 2004). Ten to fifteen years ago this number was about 25% for industrial robots and insignificant for cars.

Implementing every system from scratch does not guarantee the productivity and quality of the software systems. Software reuse has been promoted as a promising approach to improve quality and productivity. Software reuse is the use of existing software or software knowledge to construct new software (Frakes and Kang, 2005). Software reuse has been practiced and studied for a long time, and a broad reuse classification and approaches have been proposed in the area (Prieto-Diaz, 1993; Frakes and Kang, 2005). Despite the variety approaches and strategies, systematic reuse and component-based reuse are among the promising ways to improve software development efficiency and quality (Prieto-Diaz, 1993; Jacobson et al., 1997; Rothenberger et al., 2003; Ravichandran, and Marcus, 2003).

The commitment to quality is the focus of any engineering approach and in software engineering, integration of process, methods and tools for the development of quality software is also the practice (Pressman, 2005). Component-Based Software Engineering (CBSE) is a sub-discipline of software engineering, which share same strengths and weaknesses of software engineering. The general purpose of CBSE is to decrease development time and costs by creating application from reusable, easily connectable and exchangeable components. The systematic reuse can benefit from CBSE discipline by considering software component technology (Bachmann, et al., 2000) in tools, methods and processes for development of systematic reuse framework. The component-based reuse refers to bottom-up development or compositional reuse approach using reusable asset components.

CBSE solutions also receive increasing attentions in ensuring the success of ERT products (Müller et al., 2001; Crnkovic, 2004; Rastofer and Bellosa, 2001). In ERT software development, CBSE offers advantages such as software reuse, improved maintainability, and ability to easily fine-tune a real-time application’s timing properties. The rapidly changing market makes investment in CBSE for ERT systems not only viable but also essential. Encapsulated domain expertise is another characteristic of ERT systems that motivates the use of CBSE. Even small programs
in ERT systems may contain highly sophisticated algorithms, require a deep understanding in the domain and support technologies, such as the signal processing. The reuse of the domain specialized software knowledge can reduce the software development curve.

1.2 Background of the Problem

1.2.1 Some Challenges in Embedded Real-Time Software

Embedded system is a system, which contains microcomputer as a component to perform some of the requirements of that system, but the user does not see the system as a computer. Most embedded systems are real-time systems. In real-time systems timing correctness is critically important in majority of these systems as failure to meet the timing requirement can result in system malfunction or disaster. ERT systems cover a broad range of applications ranging from small–scale microwave ovens and watches to telecommunication network management and mission critical military control systems; some examples of embedded systems are given in Figure 1.1.

Figure 1.1: Examples of embedded systems
Due to the diversified nature of the ERT systems domain, as can be seen in Figure 1.1, the requirement placed on the software of ERT systems will be different from one application to another application. For example, the software requirements for consumer products, telecom switches, and avionics are quite different. The diversified nature of the ERT systems is also a reason why the software development support tools for ERT systems are much more primitive compared to the data processing domain. Tools vendors will not make much profit in a diversified ERT markets with scattered platforms and different software requirements compared to data processing market running on a uniform platform such as Windows.

In many ERT systems, requirement on low product cost is paramount. The customers are very cost sensitive, thus the developer of the hardware rarely takes the extra cost to extend the hardware resources, since the margin of profit on electronics development usually is low. Therefore the hardware is designed just enough for anticipated use but not more. Consequently, many ERT systems are resource constrained to lower the production cost and thereby increase profit. The resources here refer to power, memory consumption, computation (CPU) power, execution (CPU) time communication bandwidth etc. In their work on component technologies for Volvo vehicular industry, Möller et al. (2004) stated that resource-constrained is an important industrial requirement for automotive industries. Strong requirements for low and controlled consumption of these resource constraints create new challenges for software development in ERT systems. Coupled with the timing constraints in ERT systems, resource-constrained and timing constraints have becoming extra-functional properties which need to be addressed in the software development for many ERT systems. Many works (Ota, 2006; Hanninen and Maki-Turja, 2004; Malek et al. 2005; Cnrkovic, 2004; Nierstrasz, 2002; Hammer and Chaudron, 2001; Ommering et al. 2000) have been conducted to address the resource-constrained issue in the software development of ERT systems.

Applying component-based reuse to general ERT systems poses significant challenges to industrial software processes due to the resource-constrained and real-time requirements of the systems (Cnrkovic, 2004; Rastofer and Bellosa, 2001; Hammer and Chaudron, 2001). State-of-the-art of component-based reuse solutions such as OMG’s CORBA Component Model (CCM), Microsoft’s Component Object
Model (COM+) or Distributed COM (DCOM) family, and SUN Microsystems’ JavaBeans and Enterprise JavaBeans (EJB) are generally complex, require large resources such as memory and computation power, and are platform dependent for ERT component-based development (Lüders, 2003; Rastofer and Bellosa, 2001). Furthermore, they do not address the non-functional properties such as how much memory it consumes and timing constraints which are important in ERT systems. Consequently, a number of component technologies such as Port-based Object (PBO) (Stewart et al., 1997), Koala (Ommering et al., 2000), PErvasive COmponent Systems (PECOS) (Nierstrasz et al., 2002) and ReFlex (Wall, 2003) have been developed to address requirements of ERT software. All these ERT component models have their own unique strengths to support their nature of ERT problem domain.

1.2.2 Challenges in Autonomous Mobile Robot Software

Autonomous Mobile Robot (AMR) system is a class of ERT system with many possible applications and markets. An AMR is an autonomous system capable of traversing a terrain, performs its designated tasks, senses its environment and intelligently reacts to it. The technologies involve in the AMR system served as the basis for commercial mobile personal robots and service robots. Personal robot is used to educate, entertain or assist in the home includes for example the Lego’s Mindstorms, Sony’s Aibo robot dog and Hasbro’s Fur Real Friends robot that assists the disabled and elderly in the home. Service robots assist humans, service equipment and perform other autonomous functions in almost every industry and military applications such as unmanned ground vehicles. According to the United Nations Economic Commission and International Federation of Robotics (Kara, 2005), the personal and service robotics market roughly double between 2002 and 2005, reaching USD5.2 billion in 2005; and the number of personal and service robots sold increase ten folds between 2002 and 2005. Sales for domestic robots (vacuum cleaning, lawn mowing, window cleaning and other types) is expected to reach over 800,000 units, while sales for toy and entertainment robots will exceed one million units. As occurred in the general ERT systems, growth and competition
in the markets will force the mobile robots manufacturers to reduce the cost while at
the same time try to satisfy more functionalities demands from users. This in return
will increase the reliance on software.

AMR systems share similar requirements with ERT systems, hence inherit
the challenge in applying software reuse for general ERT systems. As the complexity
and functionality of the AMR is increased, such as adding more sensors to the robot
so as to increase its reactivity and intelligence, designing and developing software for
this type of robot can be very difficult and a challenging task. As, AMR systems and
software are becoming more and more complex (Mallet et al., 2002), the need of
highly modular software design is desirable (Seward and Garman, 1996; Messina et
al., 1999; Oreback and Christensen, 2003). On top of this, software development in
AMR systems is challenged by the diversities in terms of robot designated tasks,
robot physical size and shape, environmental interaction and implementation
platform such as real-time operating systems, hardware and communication protocol.
Furthermore, to develop AMR software involves multi-disciplines of expert
knowledge which include embedded systems, real-time software, mechanics, control
theories and artificial intelligence aspects. Thus, it is envision that component-based
reuse will be the suitable way to capture software models in this multi-disciplines
domain in order to allow reuse across applications, particularly at the early stage of
software development process. Reuse of early life-cycle of ERT systems is more
effective and it allows engineering teams to tailor reused components and designs to
fit their needs (Leveson and Weiss, 2004).

Most robot researchers are specialized in one of the area. Few robotics
research groups have the resources to build, from scratch, every component of their
robot. Nevertheless, a complete system is needed to prove any work in a special
field. Consequently, the majority of current mobile robot research platforms in use
today could not operate in a fully self-contained mode since they rely on an off-board
infrastructure, using AMR as ‘sensors with wheels’ only (Brega et al., 2000; Pont
and Siegwart, 2005). This is understandable as many researchers try to avoid
software implementation complexity in the embedded environment. However, as
argued by Brega et al. (2000) and Pont and Siegwart (2005), this off-board
processing approach is not acceptable for many applications where operating
environment size, economical aspects and safety issues do not allow off-board computing. They further emphasized that real-world AMR must be self-contained and able to meet timing constraints. Autonomy with respect to perception, energy and processing for fully self-contained autonomous decision-making is not an option but should be addressed in its full complexity already as a research topic. When on-board or embedded computation is required AMR software development is typically confronted with limited resources such as computing power and memory. Hence, meeting timing constraints becomes a problem which is present but only hidden when using off-board processing (Brega et al., 2000). Resource-constrained are especially relevant for self-contained AMR systems. Limited computing power is a typical complexity need to be confronted by on-board computation of AMR systems. With the limited computing power, a precise observation of the timing constraints of the AMR systems is a necessary to make complex robot systems more reliable. Most important, a predictable real-time performance on a robotic system is a necessary condition for guaranteeing a stable behavior of the robot (Buttazzo, 1996).

A component-based reuse solution would help in the following aspects of AMR software development (Oreback and Christensen, 2003):

i. exchange of software parts or components between robotics labs, allowing specialists to focus on their particular field,
ii. comparison of different solutions would be possible from the available components,
iii. startup in robot research can be accelerated using the available components, and
iv. speed up the transfer of research labs works in mobile robot to commercial business application.

Some of the results from component-based reuse works are hard to reuse because the solution is not coherent and simple to be used by the mechatronics and robotics engineers and researchers which are not from software engineering or computer science background (Oreback, 2004). In order to be widely accepted by robotic community, a solution for coping with systems complexity problem and software reuse for AMR should at least fulfill the following requirements (Pont and Siegwart, 2005):
i. **Embeddable and self-contained.** Typically, the AMR software or firmware is embedded in the on-board controller. A self-contained AMR system requires on-board computation and the system is typically constrained by limited processing power and memories.

ii. **Modular or component-based software.** Component modularity is to provide artifact to be reused that contain domain knowledge from different disciplines.

iii. **Portable across different platform.** Platform-independent where the component implementation does not depend on hardware, Real-Time Operating System (RTOS) or communications protocol.

iv. **Predictable real-time performance.** The abilities to predict and analyze timing are key requirements for AMR system. The reliability and reactivity of the robot behavior depend on how the robot responses to the dynamic environment events by executing a set of concurrent tasks with its timing requirements.

One of the major problems in AMR software reuse is lacking of framework that enable the domain experts to document their knowledge and plug their work in a reusable artifact and a systematic reuse process for the application robotic engineer to develop their AMR software from the reuse artifacts (Oreback, 2004; Domínguez-Brito *et al.*, 2004). Based on the literature review conducted so far, most of the existing frameworks do not address in parallel the three issues: self-contained, platform-independent and real-time predictable, for resource-constrained AMR systems in their reuse framework.

Portability of AMR components across different hardware, RTOS and communications platform is important to enable software reuse in AMR software. This can be done by minimizing the dependencies of the components with the platforms. The set of hardware component types for a robot composition is extensive. For example the input that a robot receives can be derived from a wide variety of sensors (infrared sensor, sonar sensor, video camera etc.) and each kind of sensors there are typically many variations.

According to Frakes and Kang (2005) there are nine active research areas under software reuse category. They are: business and finance; measurement and experimentation; componentry; domain engineering; programming languages;
librarians; architectures; generative methods; and reliability and safety. From these nine areas, three research areas were identified to be important in developing a framework for AMR component-based reuse, i.e. componentry, domain engineering and reliability. Componentry concerns with CBSE technologies to provide important platform or framework on which reusable AMR components can be developed and applications can be created by integrating the components. Domain engineering enables building of AMR system variants repeatedly within the same domain. The ability to predict the reliability of the integrated AMR components through timing performance analysis is one of key success of software reuse in AMR software. The research problems in these three software reuse research areas need to be solved before the AMR framework can be developed.

1.3 Statement of the Problem

In order to develop the component-based reuse framework for AMR domain there are numbers of software reuse research problems need to be solved in the three research areas: componentry, domain engineering and reliability, as identified above. The research problems need to be answered in determining how to support robotics programmers and engineers in creating reusable artifacts, and in the application of the reuse artifact. The problems to be solved are:

i. What should be made reusable in AMR software?
ii. What are the methods to specify reuse components and how to use the reuse components in developing AMR software?
iii. How to predict the reliability of AMR component composition?

Applying component-based reuse to ERT mobile systems poses a significant challenge due to the resource-constrained and real-time requirements of ERT systems and the degree of diversity in AMR systems. The existing works on component-based reuse have not addressed both ERT and AMR requirements concurrently, and proposed solution frequently hard to be used by the domain experts. To obtain an acceptable component-based reuse solution for ERT mobile
robot systems the solution must have a systematic reuse framework to enable the robotic engineers to plug their work in the framework with consideration to the three basic requirements: self-contained, platform-independent and real-time predictable AMR systems. Thus, the research goal is:

To develop a framework for component-based systematic reuse of autonomous mobile robots (AMR) software which emphasized on requirements to be self-contained, platform-independent and real-time predictable.

Software framework is a general word whose meaning depends heavily on the context in software engineering discipline. There are different types of frameworks in various software discipline literature which include conceptual framework (Schneider, 1999), architecture framework (Zachman, 1987), application framework (Fayad and Schmidt, 1997), development framework (Bass et al., 1994) and component framework (Bachmann et al., 2000). The development framework as proposed by Bass et al. (1994) work is suitable to support systematic reuse in AMR CBSE since the development framework translates application-specific system specifications using methods that familiar to developers and optimal for the application. The development framework can be used to emphasize the optimization of the three AMR requirements: self-contained, platform-independent and real-time predictable. The framework covers at least three phases of software development: specification, design and implementation and the framework highlights the software techniques, strategies and tools in order to achieve the optimizations aimed in using the framework.

1.4 Objectives

The main objectives of this research work reported in this thesis are:

1. To determine the reusable components of AMR software and framework to document them.
2. To assess and evaluate the applicability of the existing ERT component models and their implementation frameworks for the use in AMR software development.

3. To determine how existing real-time theories can be used for predicting timing performance of an AMR components composition.

4. To develop a CBSE framework that consists of AMR reuse artifacts, ERT component model and timing analysis approach.

5. To experimentally evaluate effectiveness of the proposed framework on some real existing AMR systems, and to measure some of the reuse qualities produce by the framework.

1.5 Scope of the Study

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

- This research focuses on resource-constrained AMR applications.
- The AMR reusable asset documented in this thesis is assumed to be a starting set of reuse components. These components will continue to evolve as the domain expert from academia and industry involve in the assets development.
- The modeling of the reuse components will only consider structural modeling and real-time behavior modeling in order to illustrate the component-level modeling.
- The implementations on the existing AMR systems in this work are aimed to prove experimentally the proposed framework and the applicability of the elements in the framework.
- This research is focusing in reactive layer of hybrid model, since; software at reactive layer is typically constrained by limited resources and real-time
requirements.

1.6 Thesis Outline

Chapter 2 discusses strategies and challenges in software reuse and CBSE in ERT generally and AMR specifically. The state-of-the-art of software reuse and CBSE in AMR software is reviewed in Chapter 2.

In Chapter 3, the research methodology conducted in achieving the objectives of this research is presented. Specifications two real AMR systems which were used in case studies, implementation of embedded software and verifications of techniques proposed are described in this chapter.

The three main elements supporting the developed component-based reuse framework in this thesis are described in details in Chapter 4 through 6. Chapter 4 describes the component-based AMR analysis patterns. The evaluation of a component model and the extension to the component model to support the required AMR software development environment are discussed in Chapter 5. In Chapter 6 a detailed timing analysis based on component-based reuse for AMR timing prediction at design phase is proposed.

The integration of the elements proposed in Chapter 4 through 6 to form the developed CBSE framework will be discussed in detail in Chapter 7. The application of the framework to develop AMR software will be illustrated in Chapter 7. To validate some of the reuse qualities resulting from the proposed framework, measurement of reusability and amount of reuse qualities were conducted on the developed software. The validation process and results are discussed in Chapter 8. Finally in Chapter 9, the conclusions and some suggestions for future work are given.
REFERENCES


