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

Perlaksanaan guna-semula perisian dalam sistem terbenam masa-nyata (ERT) merupakan cabaran penting kepada proses perisian di industri. Ini disebabkan oleh sumber yang terhad dan keperluan masa-nyata sistem ERT. Sistem robot bergerak berautonomi (AMR) mewarisi cabaran ini kerana ia adalah daripada kelas sistem TMN. Cabaran ini ditambah pula dengan kepelbagaian dalam sistem robot dari segi saiz, bentuk, interaksi dengan persekitaran dan pelantar perlaksanaan. Oleh itu, guna-semula berasaskan-komponen dilihat sebagai satu cara yang sesuai untuk menggalakkan guna-semula perisian dalam sistem ERT dengan mempertimbangkan tiga keperluan umum AMR iaitu pemprosesan dalaman, ketidakbergantungan kepada pelantar perlaksanaan, dan masa-nyata yang boleh diramal. Dalam tesis ini, satu rangka kerja guna-semula berasaskan-komponen bagi perisian RBB telah dibangunkan untuk membolehkan guna-semula yang sistematik melalui kejuruteraan perisian berasaskan-komponen. Rangka kerja ini bertujuan untuk memberi panduan dalam strategi guna-semula perisian dalam aplikasi RBB. Rangka kerja ini mengandungi empat unsur utama: corak analisis berasaskan-komponen AMR, model komponen yang diubah-suai, pendekatan analisis pemasaan berasaskan-komponen, dan rangka kerja pengaturcaraan berorientasikan-komponen. Hasil perlaksanaan rangka kerja ini dalam membangunkan perisian RBB menunjukkan strategi dan proses yang dicadangkan dalam rangka kerja ini boleh memenuhi keperluan umum RBB. Untuk menentukan keberkesanan pendekatan guna-semula pada rangka kerja ini, pengukuran kebolehgunaan-semula komponen dan kadar guna-semula dibuat dengan menggunakan metrik perisian. Hasil pengukuran menunjukkan kadar kebolehgunaan-semula adalah tinggi pada komponen yang dikehendaki dan kadar guna-semula mencapai sehingga 74% dalam RBB yang diuji. Hasil perlaksanaan dan pengukuran guna-semula menunjukkan rangka kerja tersebut menggalakkan guna-semula yang sistematik dan berkualiti.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ERT	-	Embedded Real-Time
AMR	-	Autonomous Mobile Robot
ART-ML	-	Architecture and Real-Time behavior Modeling Language
BBC	-	Behavior-Based Control
CBD	-	Component-Based software Development
CBSE	-	Component-Based Software Engineering
CCM	-	CORBA Component Model
CLARAty	-	Coupled Layered Architecture for Robotic Autonomy
CoCo	-	Component Composition
COM	-	Component Object Model
CoolBOT	-	Component-Oriented Programming Framework for Robotics
COP	-	Component-Oriented programming
CORBA	-	Common Object Request Broker Architecture
COTS	-	Commercial off-the-shelf
DC	-	Direct-Current
DCOM	-	Distributed Component Object Model
DX	-	Data Exchange
EDF	-	Earliest Deadline First
EJB	-	Enterprise JavaBeans
FBD	-	Function Block Diagram
HAL	-	Hardware Abstraction Layer
HLRC	-	High Level Robot Control
HRI	-	Human-Robot Interface
IDL	-	Interface Definition Language
IR	-	Infra-Red
JVM	-	Java Virtual Machine

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

VRF	-	Virtual Robot Framework
WCET	-	Worst-Case Execution Times
XML	-	eXtensible Markup Language

LIST OF APPENDICES

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.

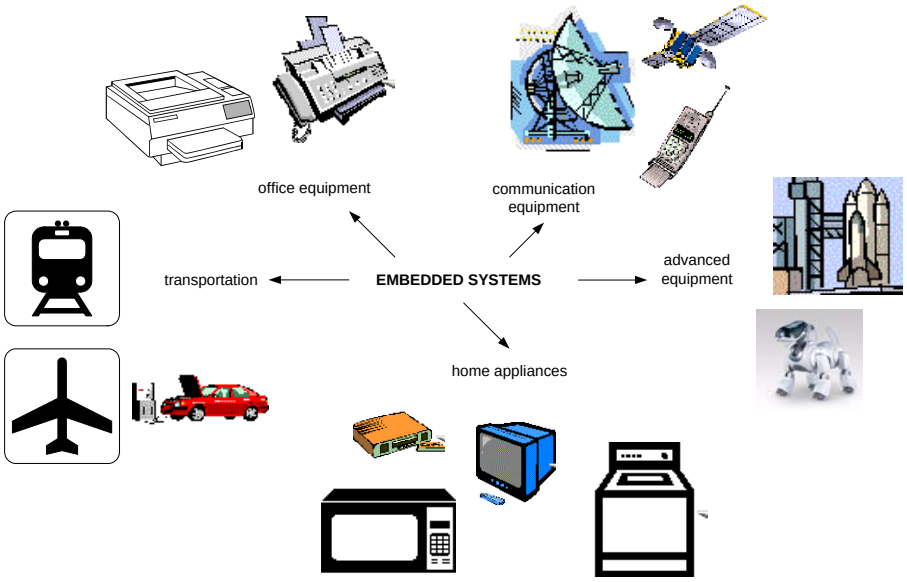

Figure 1.1: Examples of embedded systems

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 compare 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 (Crnkovic, 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 depends 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 problem 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;

libraries; 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

Åkerholm, M. and Fredriksson, J. (2004). *A Sample of Component Technologies for Embedded Systems*. Mälardalen Research and Technology Centre (MRTC), Mälardalen University: Technical Report.

Alami, R., Chatila, R., Fleury, S., Ghallab, M. and Ingrand, F. (1998). Architecture for Autonomy. *Journal of Robotics Research.* 17(4): 315-337.

Amin S. H. M., Mamat R., Khessal N. O. and Rudas I. J. (1999) Intelligent Behaviour-Based Climbing-Robot. *Proceeding of IEEE International Conference on Intelligent Engineering System 1999 (INES'99)*. November 1999. Slovak Republic. 78-92.

Amnell, T., E. Fersman, L. Mokrushin, P. Pettersson, and Yi, W. (2003). TIMES: a Tool for Schedulability Analysis and Code Generation of Real-Time Systems. in *Proceedings of the 1st International Workshop on Formal Modeling and Analysis of Timed Systems FORMATS 2003*. September 6-7. Marseille, France:IEEE, 60-72.

Audsley, N., Burns, A., Richardson, M., Tindell, K. and Wellings, A. (1993). Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*. 8(5): 284-292.

Bachmann, F., Bass, L., Buhman, S., Comella-Dorda L. S., Seacord, R. C. and Wallnau, K. C. (2000). *Technical Concept of Component-Based Software Engineering*. Software Engineering Institute, Carnegie Mellon University: Technical Report CMU/SEI-2000-TR-008.

Bass, J. M., Browne, A. R., Hajji, M. S., Marriott, D. G., Croll, P. R. and Fleming, P. J. (1994). Automating the Development of Distributed Control Software. *IEEE Parallel and Distributed Technology*. 9-19.

Bass, L., Clements, P. and Kazman, R. (1998). *Software Architecture in Practice*, Reading: Addison-Wesley.

Blum, S. (2001). Towards a Component-based System Architecture for Autonomous Mobile Robots. *Proceedings of IASTED International Conference Robotics and Applications (RA'01)*. November 19-22. Tampa, Florida, 220-225.

Bouyssounouse, B. and Sifakis, J. (2005). Embedded Systems Design: The ARTIST Roadmap for Research and Development. In: Bouyssounouse, B. and Sifakis, J. *Lecture Notes in Computer Science*. Volume 3436/2005, Berlin, Heidelberg:Springer-Verlag GmbH. 160-194.

Braunl, T. (2003). E*mbedded Robotics: Mobile Robot Design and Applications with Embedded Systems*. New York: Springer-Verlag.

Brega, R., Tomatis, N. and Arras, K.O. (2000). The Need for Autonomy and Real-Time in Mobile Robotics: A Case Study of XO/2 and Pygmalion. Proceedings *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, Volume 2. October 30- November 5. Takamatsu, Japan:IEEE, 1422 – 1427.

Brooks, A., Kaupp, T., Makarenko, A., Orebäck, A. and Williams, S. (2005) Towards Component-Based Robotics. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*. August 2-6. Edmonton, Alberta: IEEE, 163 – 168.

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

Brown, A.W. and Wallnau, K.C. (1998). The Current State of Component Based Software Engineering. *IEEE Software.* 15(5): 37 –46.

Bruyninckx, H. (2001). Open Robot Control Software: the OROCOS Project. *Proceedings IEEE International Conference on Robotics and Automation (ICRA) 2001*, Volume 3. May 21-26. Seoul, Korea: IEEE, 2523 – 2528.

Buschmann, F, Meunier, R., Rohnert, H. Sommerlad, P. and Stal, M. (1996). *Pattern-Oriented Software Architecture A System of Pattern*. England: John Wiley and Sons.

Buttazzo, G. C. (1996). Real-time Issues in Advanced Robotics Applications. *Proceedings of the 8th IEEE Euromicro Workshop on Real-time Systems*. June 1996. L'Aquila, Italy:IEEE, 133-138.

Buttazzo, G. C. (2005). Rate Monotonic vs. EDF: Judgment Day. Real-Time Systems Journal, 29: 5–26.

Cheesman, J. and Daniels, J. (2001). *UML Components*. Boston: Addison-Wesley.

Chikofsky, E. J. and Cross II, J. H. (1990). Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, **7**(1), 13-17.

Coad, P., North, D. and Mayfield, M. (1997). *Object Models: Strategies, Patterns, and Applications*, New Jersey: Yourdon Press.

Creps, R. E. (1992). The STARS Conceptual Framework for Reuse Processes. *Software Engineering Notes for The Fifth Annual Workshop on Institutionalizing Software Reuse (WlSB'92)*. 18(2): 74-85.

Crnkovic, I. (2004). Component-based Approach for Embedded Systems, *Proceedings of the Ninth International Workshop on Component-oriented Programming*, Session 4 – Application of CBSE. June 14–18. Oslo, Norway.

Crnkovic, I., and Larsson, M. (2002). *Building Reliable Component-based Software Systems*. Norwood: Artech House, 2002.

Crnkovic, I., Hnich, B. Jonsson, T. and Kiziltan, Z. (2002). Specification, Implementation, and Deployment of COMPONENTS. *Communication of the ACM*. 45(10): 35-40.

Cybulski, J. L., Neal, R. D., Kram, A. and Allen J. C. (1998). Reuse of Early Life-Cycle Artifacts: Workproducts, Methods and Tools. *Annals of Software Engineering*. 5: 227–251.

Domínguez-Brito, A. C. (2003). CoolBOT: a Component-Oriented Programming Framework for Robotics. Univiversity de Las Palmas de Gran Canaria: Ph.D. Thesis.

Dominguez-Brito, A.C., Hernandez-Sosa, D., Isern-Gonzalez, J., and Cabrera-Gamez, J. (2004). Component software in robotics. *Proceedings of the 2nd International IEEE Conference Intelligent Systems 2004*, Volume 2. 22-24 June 2004. Varna, Bulgaria: IEEE, 560 – 565.

Douglass, B. P. (2002). *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. Boston: Addison Wesley.

Douglass, B. P. (2004). *Real-Time UML*. Boston: Addison Wesley. 2004.

Fenton, N. E. and Pfleeger, S.L. (1997). Software Metrics: A Rigorous and Practical Approach. 2nd Edition. Boston, Mass: PWS.

Fernandez, J.A. and Gonzalez, J. (1998). NEXUS: a flexible, efficient and robust framework for integrating software components of a robotic system, *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 1. May 16-20. Leuven, Belgium:IEEE, 524 – 529.

Finkelstein, A. and Kramer, J. (2000) Software Engineering: A Roadmap**.** *Proceedings of the conference on The future of Software Engineering*. May 2000. Limerick, Ireland:ACM, 5-22.

Fire Marshall Bill (2004). at www.dragonflyhollow.org/matt/robots/firemarshallbill/ available August 2004.

Fowler, M. (1997). *Analysis Patterns: Reusable Object Models*. Boston: Addison-Wesley.

Frakes, W. B., and Kang, K. (2005). Software Reuse Research: Status and Future. *IEEE Transaction On Software Engineering*. 31(7): 529-536.

Frakes, W., and Terry C. (1996). Software Reuse: Metrics and Models. *ACM Computing Surveys (CSUR)*. 28(2): 415 – 435.

Frakes, W.B. and Isoda, S. (1994). Success factors of systematic reuse. *IEEE Software*. 11(5):14 – 19.

Fujita, M. and Kageyama, K. (1997). An Open Architecture for Robot Entertainment. *Proceedings of the First International Conference on Autonomous Agents*, February 5-8. Marina Del Rey, USA:ACM, 435-442.

Gamma, J., Helm, R., Johnson, R. and Vlissides, J. (1995). *Design Patterns: Elements of Reuse Object-Oriented Software*. Reading: Addison-Wesley.

Genssler, T. Christoph., A., Schulz B, Winter, M., Stich, C. M., Zeidler C., Muller, P., Stelter, A., Nierstrasz, O., Ducasse, S., Arevalo, G., Wuyts, R., Liang, P., Schonhage, B., van den Born, R. (2002). *PECOS in a Nutshell*. PECOS Project: Technical Handbook.

George, R. and Kanayama, Y. (1996). A Rate-Monotonic Scheduler for Real-time Control of Autonomous Robots. *Proceeding of the 1996 IEEE International Conference on Robotics and Automation*. April 22-28. Minneapolis, Minnesota:IEEE, 2804-2809.

Geyer-Schulz, A. and Hahsler, M. (2002). Software reuse with analysis patterns. *Proceedings of the 8th Association for Information Systems (AMCIS)*. August 9-11. Dallas, Taxes, 1156-1165.

Goulão, M. and Brito-e-Abreu, F. (2004). Software Components Evaluation: an Overview. *Proceeding of the 5th Information Systems Portuguese Association Conference (CAPSI'2004)*. November. Lisbon, Portugal.

Graves, A. R., and Czarnecki, C. (2000). Design Patterns for Behaviour-Based Robotics. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Human*. 30(1): 36-41.

Gu, Z. and He, Z. (2005). "Real-Time Scheduling Techniques for Implementation Synthesis from Component-Based Software Models", In: Heineman, G. T., Crnkovic, I., Schmidt, H. W., Stafford, J. A., Szyperski, C. and Wallnau, K. *Component-Based Software Engineering 2005, Lecture Notes in Computer Science 3489*. Berlin, Heidelberg:Springer-Verlag, 235–250.

Hänninen, K. and Mäki-Turja J. (2004). *Component technology in Resource Constrained Embedded Real-Time Systems*. Mälardalen Research and Technology Centre (MRTC), Mälardalen University: Technical Report.

Hammer, D.K. and Chaudron, M.R.V. (2001). Component-Based Software Engineering for Resource-Constraint Systems: What are the Needs?. *Proceedings of the Sixth International Workshop on Object-Oriented Real-Time Dependable Systems*. January 8-10. Rome, Italy:IEEE, 91 – 94.

Hassan, H., Simo, J. and Crespo A. (2001). Flexible Real-Time Mobile Robotic Architecture Based on Behavioural Models. *Journal of Engineering Applications of Artificial Intelligence*. 14(5): 685-702.

Heinenam, G. T. and Councill, W. T. (2001). *Component-Based Software Engineering*. Canada: Addison-Wesley.

Hissam, S. and Ivers, J. (2002). *PECT Infrastructure: A Rough Sketch*. Software Engineering Institute, Carnegie Mellon University: Technical Note CMU/SEI-2002-TN-033.

Hissam, S., Moreno, G. Stafford, J. and Wallnau. K. (2003). Enabling Predictable Assembly. *The Journal of Systems and Software*. 65: 185-198.

Hissam, S. A., Hudak J., Ivers J., Klein M., Larsson M., Moreno G. A., Northrop L., Plakosh D., Stafford J., Wallnau K. C., and Wood W. (2003b). *Predictable Assembly of Substation Automation Systems: An Experience Report, Second Edition*, Software Engineering Institute, Carnegie Mellon University: Technical Report CMU/SEI-2002-TR- 031.

IBM Rational Software Corporation, Rational Rose Real-Time, Available from http:www-8.ibm.com/developerworks/rational/products/rosetechnicaldeveloper.

iRobot Corporation, (2002). *Mobility Robot Integration Software User's Guide*, Jaffrey: iRobot Corporation.

Isovic, D. and Norstrom, C. (2002). Components in Real-Time Systems. *Proceeding of the 8th International Conference on Real-Time Computing Systems and Applications (RTCSA'2002)*. March 18-20. Tokyo, Japan.

Jacobson, I., Griss, M. and Jansson, P. (1997). *Software Reuse: Architecture, Process and Organization for Business Success*. New York: ACM Press/Addison-Wesley.

Jawawi D. N. A. (2000). The Development of Real-time Control Firmware for A Wall Climbing Robot - A Small-Scale Embedded Hard Real-time System. Universiti Teknologi Malaysia: Master Thesis.

Jones, L.J., Seiger, B.A., and Flynn, A.M. (1999). *Mobile Robots Inspiration to Implementation*. Second edition. Natick: A K Peters.

Kara, D. (2005). Sizing and Seizing the Robotics Opportunity. Available from http://www.robonexus.com/roboticsmarket.htm.

Kirchner L. (2005). Carnegie Mellon University in Qatar Launches First International Botball Robotics Club**.** *Carnegie Mellon Press Release* at http://www.cmu.edu/PR/releases05/050516_botball.html available May 16, 2005.

Klien, M., Ralya, T., Pollak, B. and Obenza, R. (1993). *A Practitioner's Handbook for Real-time Analysis*, Massachusetts: Kluwer Academic Publisher.

Konrad S., Cheng B. H. C. and Campbell L. A. (2004). Object Analysis Patterns for Embedded Systems. *IEEE Transactions on Software Engineering*. 30(12): 970-991.

Krueger, C. W. (1992). Software Reuse. *ACM Computing Surveys*. 24(2): 131-183.

Kwong T. C., Amin S. H. M., Mamat R. and Rudas I. J. (2004). An Intelligent Voting Technique in Behavior-based Autonomous Mobile Robot for Goal-Directed Navigation. In: *Current Trends in Artificial Intelligence and Application*. Kuala Lumpur: UMS. 249-256.

Labrosse, J. J. (1999). *MicroC/OS-II The Real-Time Kernel*. 2nd ed. Berkeley: R&D Books.

Larsson, M. (2004). *Predicting Quality Attributes in Component-based Software Systems*. Mälardalen University: Ph.D. Thesis.

Lehoczky, J., Sha, L. and Ding, Y. (1989). The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. *Proceedings of Real Time Systems Symposium*. December. Santa Monica, California:IEEE, 66 – 171.

Lehoczky, J.P. (1990). Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. *Proceedings of the 11th Real-Time Systems Symposium*. 5-7 December, 201 – 209. Lake Buena Vista, Florida:IEEE, 201-213.

Leveson, N. G. and Weiss, K. A. (2004). Making Embedded Software Reuse Practical and Safe. *Proceedings of the 12th ACM SIGSOFT twelfth International Symposium on Foundations of Software Engineering*. November 2-4. Newport Beach, California:ACM, 171 – 178.

Lim C. S. (2004). *Development and Implementation of Internet-Based Task-Oriented Telerobotic System.* Universiti Teknologi Malaysia, Malaysia: Master Thesis.

Lüders, F. (2003). Adopting a Software Component Model in Real-Time Systems Development. *Proceedings 28th Annual NASA Goddard Software Engineering Workshop*. December 3-4. Greenbelt, Maryland:ACM, 114 – 119.

Lui, C. L. and Layland, J. (1973). Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*. 20(1): 46-61.

Majid A. M., Amin S. H. M. and Mamat R. (2001), Development of the Internet Interface for Leg Motion Control of A Mobile Robot, *Proceedings of the First International Conference on Mechatronics (ICOM'01)*. 12th – 13th February 2001. Kuala Lumpur. 665-675.

Malek, S., Mikic-Rakic M. and Medvidovic N. (2005). A Style-Aware Architectural Middleware for Resource-Constrained, Distributed Systems. *IEEE Transactions on Software Engineering*. 31(3). 256-272.

Mallet, A. Fleury, S. and Bruyninckx, H. (2002). A Specification of Generic Robotics Software Components: Future Evolutions of GenoM in the Orocos Context. *Procceding of the IEEE International Conference on Intelligent Robots and System,* Vol. 3. September 30 - October 4. EPFL, Switzerland:IEEE, 2292 – 2297.

Masse, J., Kim, S. and Hong, S. (2003). Tool Set Implementation for Scenario-based Multithreading of UML-RT Models and Experimental Validation. *IEEE*

Proceeding of Real-Time and Embedded Technology and Applications Symposium. May 27–30. Toronto, Canada:IEEE, 70–77.

Messina, E., Horst, J., Kramer, T., Huang, H. and Michaloski, J. (1999). Component Specifications for Robotics Integration, *Autonomous Robots*, 6: 247-264.

Minoura, T., Pargaonkar, S. and Rehfuss, K. (1993). Structural Active Object Systems for Simulation. *Proceedings of the Eighth Annual Conference on Object-oriented Programming Systems, Languages, and Applications*. September 26 - October 1. Washington: SIGPLAN Notices, 338-355.

Mohd. Ridzuan Bin Ahmad (2003). *Development of Reactive Control Algorithm for Multi-Agent Robotics System in Cooperative Task Achievement.* Universiti Teknologi Malaysia: M.Eng. Thesis.

Möller, A., Åkerholm, M., Fredriksson, J. and Nolin, M. (2004). Evaluation of Component Technologies with Respect to Industrial Requirements. *Proceedings of the 30th EUROMICRO Conference on Component-Based Software Engineering Track*. September 1 – 3. Rennes, France, 56-63.

Müller, P. Stich, C. and Zeidler, C. (2001). Components @ Work: Component Technology for Embedded Systems. *Proceedings of the 27th Euromicro Conference 2001: A Net Odyssey*. 4-6 September. Warsaw, Poland:IEEE, 64 – 71.

Nelson, M. L. (1999). A Design Pattern for Autonomous Vehicle Software Control Architectures. *Proceeding of 23rd International Conference on Computer Software and Applications*, October 27-29. Phoenix, AZ:IEEE, 172-177.

Nesnas, I.A., Volpe, R., Estlin, T., Das, H., Petras, R. and Mutz, D. (2001). Toward Developing Reusable Software Components for Robotic Applications. *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. October 29 – November 3, Maui, Hawaii:IEEE, 2375 - 2383.

Nesnas, I.A., Wright, A., Bajracharya, M., Simmons, R., Estlin, T., Kim W. S. (2003). CLARAty: An Architecture for Reusable Robotic Software. *Proceedings of SPIE Aerosense Conference*, Volume 5083 - Unmanned Ground Vehicle Technology V. April 21-25. Orlando, FL, 253-264.

Nierstrasz, O., Arévalo, G., Ducasse, S., Wuyts, R., Black, A., Müller, P., Zeidler, C., Genssler, T., van den Born. R. (2002). A Component Model for Field Devices. *Proceedings First International IFIP/ACM Working Conference on*

Component Deployment. June 20-21. Berlin: Springer-Verlag Heidelberg, 200-209.

Ota, J. (2006). Search Methodology with Goal State Optimization Considering Computational Resource Constraints. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2006*. May 15-19. Orlando, Florida: IEEE, 2023 – 2028

OMG Object Management Group (2003). *Reusable Asset Specification: Version 2.2*. Needham: Object Management Group.

OMG Object Management Group (2002). *Unified Modeling Language Specifications. Version 1.3*. Needham: Object Management Group.

Ommering, R., Linden, F., Kramer, J. and Magee, J. (2000). The Koala component model for consumer electronics software. *IEEE Computer*, 33(3): 78 –85.

Orebäck A. (2004). *A Component Framework for Autonomous Mobile Robots*. KTH Stockholm: Ph.D. Thesis.

Orebäck, A., and Christensen, H. I. (2003). Evaluation of Architecture for Mobile Robotics. *Autonomous Robots*. 14: 33-49.

Paradigm Systems (2000). *Paradigm C++ Reference Manual Version 5.0*, Endwell.

Pont, F., and Siegwart, R. (2005). A Real-Time Software Framework for Indoor Navigation. *Proceedings of the IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (IROS)*. August 2-6. Edmonton, Canada: IEEE, 2085 - 2090.

Pont, M. J. and Banner, M. P. (2004). Designing embedded systems using patterns: A case study**,** *Journal of Systems and Software*, 71(3): 201-213.

Purnomo D. S. (2004). *Active Force Control of A Nonholonomic Wheeled Mobile Robot*. Universiti Teknologi Malaysia, Malaysia: Master Thesis.

Poulin, J. S. (1997). Measuring Software are Reuse: Principles, Practices, and EconomicModels. Reading: Addison-Wesley.

Pressman, R. S. (2005). Software Engineering: A Practitioner's Approach. 6th ed. New York: McGraw-Hill Higher Education.

Prieto-Diaz, R. (1993). Status Report: Software Reusability. *IEEE Software*. 10(3). 61-66.

Rastofer, U. and Bellosa, F. (2001). Component-based Software Engineering for Distributed Embedded Real-time Systems. *IEE Proceeding- Software*, 148(3): 99-103.

Ravichandran, T. and Marcus, A. R. (2003). Software Reuse Strategies and Component Markets. *Communications of the ACM*. 46(8): 109-114.

Redwine, S. T. and Riddle, W. E. (1985). Software Technology Maturation. *Proceedings of the 8th international Conference on Software Engineering Conference*. August 28-30. London, England: IEEE, 189 - 200.

Riehle, D. and Zullighoven, H. (1996). Understanding and Using Patterns in Software Development. *Theory and Practice of Object Systems,* 2(1): 33-13.

Rine, D. C. and Nada, N. (2000). An empirical study of a software reuse reference model. *Information and Software Technology*. 42(1): 47-65.

Rivero, D. M., Khamis A. Rodriguez F. J. and Salichs M. A. (2003). A Patterns-Oriented Framework for the Development of Automatic and Deliberative Skills for Mobile Robots. *Proceedings of the 11th International Conference on Advanced Robotics*. June 30 - July 3. Portugal:IEEE, 3284 - 3289.

Rothenberger, M. A. and Hershauer, J. C. (1999). A Software Reuse Measure: Monitoring an Enterprise-Level Model Driven Development Process. *Journal of Information & Management*. 35(5): 283-293.

Rothenberger, M.A., Dooley, K.J., Kulkarni, U.R., and Nada, N. (2003). Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices. *IEEE Transactions on Software Engineering.* 29(9). 825 – 837.

Saksena, M. and Karvelas, P. (2000). Designing for Schedulability: Integrating Schedulability Analysis with Object-Oriented Design. *IEEE Proceeding of EuroMicro Conference on Real-Time Systems*. June 19 - 21. Stockholm, Sweden:IEEE, 101–108.

Schneider J. (1999). Components, Scripts, and Glue: A Conceptual Framework for Software Composition. der Universit¨at Bern: Ph. D. Thesis.

Selic, B. and Ward, P. T. (1996). “The Challenges of Real-time Software Design.” *Embedded Systems Programming*, 9.(11).

Seward, D. W. and Garman, A. (1996). The Software Development Process for an Intelligent Robot. *IEEE Computing and Control Engineering Journal*, 7(2): 86 – 92.

Sha L., Abdelzaher, T., Årzén, K., Cervin, A., Baker, T., Burns, A. Buttazzo, G., Caccamo, M., Lehoczky J. and Mok, A. K. (2004). Real-Time Scheduling Theory: A Historical Perspective. *Real-Time Systems Journal*. 28: 101-155.

Sha, L., Rajkumar, R. and Lehoczky, J. P. (1990). Priority Inhetitance Protocols: An Approach of Real-Time Synchronization. *IEEE Transaction Computer*. 39(9): 175-185.

Shaw, M. (2002). What Makes Good Research in Software Engineering?. *International Journal of Software Tools for Technology Transfer*. 4(1): 1-7.

Shi, J., Goddard, S., Lal, A., Dumpert, J. and Farritor, S. (2005). Global Control of Robotic Highway Safety Markers: A Real-time Solution. *Real-Time Systems Journal*. 29: 183-204.

Sindre G. and Conradi R. (1995). The REBOOT Approach to Software Reuse. *Journal of Systems Software*. 30: 201-212.

Smith, G., Smith, R. and Wardhani, A. (2005). Software Reuse across Robotic Platforms: Limiting the Effects of Diversity. *Proceedings of the Australian Software Engineering Conference*. March 31 - April 1. Brisbane, Australia:IEEE, 252 – 261.

Stankovic, J. A. and Ramamritham, K. (1990). What if Predictability for Real-Time Systems. *Real-Time Systems Journal*. 2: 247–254.

Stewart, D. B., Volpe, R. A., and Khosla, P. K. (1997). Design of Dynamically Reconfigurable Real-time Software Using Port-Based Objects, *IEEE Transaction on Software Engineering*, 23(12): 759 –776.

Su E. L. M., Amin S. H. M., Mamat R. and Yeong C. F. (2005). Room Recognition for Mobile Robot Using Appearance-Based Method. *The Proceeding of 9th International Conference on Mechatronics Technology 2005*. 5-8 December 2005. Kuala Lumpur. ICMT-185.

Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming*. 2nd Edition. London:Addison-Wesley / ACM Press.

Thrun S. (2000). Towards Programming Tools for Robots That Integrate Probabilistic Computation and Learning. *Proceedings of the IEEE International Conference on Robotics and Automation*. April 2000. San Francisco, CA: IEEE, 306-312.

Volpe R., Nesnas I.A.D., Estlin T., Mutz D., Petras R., and Das H. (2000). *CLARAty: Coupled Layer Architecture for Robotic Autonomy*. Jet Propulsion Laboratory, California Institute of Technology: Technical Report D-19975.

Wall, A. (2003). *Architectural Modeling and Analysis of Complex Real-Time Systems*. Mälardalen University: Ph.D. Thesis.

Wang, A. J. A. and Qian, K. (2005). *Component-Oriented Programming*. New Jersey: John Wiley and Sons.

Washizaki, H., Yamamoto, H. and Fukazawa, Y. (2003). "A Metrics Suite for Measuring Reusability of Software Components", *Proceedings of the Ninth International Software Metrics Symposium*. September 3-5. Sydney, Australia:IEEE, 211 – 223.

Winn, T. and Calder, I. (2002). This a Pattern?. *IEEE Software*. 59-66.

WinAVR (2006). Available from http://sourceforge.net/projects/winavr/

Wuyts, R., Ducasse, S. and Nierstrasz, O. (2005). A Data-centric Approach to Composing Embedded, Real-time Software Components. *The Journal of Systems and Software*, 74: 25-34.

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

Yeong C. F., Amin S. H. M., Mamat R., Fisal N. and Su E. L. M. (2005). Development and Evaluation of Various Modes of Human Robot Interface for Mobile Robot. *The Proceeding of 9th International Conference on Mechatronics Technology 2005.* 5-8 December 2005. Kuala Lumpur. ICMT-184.

Zachman J. A. (1987). A Framework for Information Systems Architecture. IBM Systems Journal. 26(3). 276-292.