# A Review of Real-Time Software Engineering Methodologies for Developing a Wall-Climbing Robot Control Firmware

Dyg. Norhayati Abg. Jawawi, Safaai Deris
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Johor Bahru, Malaysia
email : dayang@fsksm.utm.my

Rosbi Mamat
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 Johor Bahru, Malaysia

**Abstract:** Designing and developing software for autonomous robot control system is a challenging task. Issues related to real-time control, embedded system and artificial intelligence are involved in the software development process. This type of software must be developed with proper software methodology or well-define development process in order to increase the productivity and quality of the software design and software products. This paper presents a review of two real-time software development methodologies and compares their suitability for developing real-time control software for a wall-climbing robot under development at Universiti Teknologi Malaysia (UTM).

**Keywords**
Real-time systems, embedded systems, real-time software methodology, autonomous climbing robot

## I. INTRODUCTION

Climbing robots belong to the class of mobile robot. A Wall-Climbing Robot (WCR) climbs up and down the wall, performs its designated tasks, senses its environment and intelligently reacts to it. As the complexity and functionality of the robot is increased, such as adding more sensors to the robot so as to increase its reactivity and intelligence, designing and developing control software for this type of robot can be very difficult and a challenging task.

Issues related to real-time control, embedded system and artificial intelligence are involved in the climbing robot software development process. This type of software must be developed with proper software methodology or well-defined development process. Typically, the software or firmware is embedded in the onboard controller. To provide intelligence and reactive action, the robot firmware must sense its environment with multiple sensors and process the information and taking actions in real-time.

In a real-time system such as a climbing robot, the correctness of the system depends not only on the logical results, but also on the time at which the results are produce. A real-time system is inherently concurrent and multitasking since it has to react to and process numerous events simultaneously. Therefore real-time software differs from the traditional data processing software in that they are constrained by non-functional requirements such as timing and dependability.

In order to increase the software productivity, maintainability and flexibility in developing real-time robot software, proper software engineering need to be considered. Proper software engineering methodology ensures the software development process is manageable, and that reliable and correct program is constructed. The aim of this paper is to review some of the Real-Time Software Engineering Methodologies (RTSEM) and critically assess the suitability of these RTSEM for developing the UTM WCR control firmware.

This paper is organised as follows. Section II discussed the WCR controller hardware, which is under development at the Universiti Teknologi Malaysia (UTM). In Section III, two popular RTSEM main attributes are studied and applied to the WCR control firmware design. Section IV presents some results comparing and assessing the suitability of these RTSEM for developing the WCR control firmware. Finally, the conclusion is presented in Section V.

## II. THE WALL-CLIMBING ROBOT CONTROLLER

The UTM wall-climbing robot consists of a body and four similar jointed legs. The body carries the embedded controller and it's associated electronics, and others load. Each leg has three joints, which will give a three-degree of freedom movement for each leg. Each joint is move by a direct-current (DC) motor and position sensors measure the angles of movement for each joint. At the tips of each leg there is a suction pad which will stick the robot on the wall. Pressure sensors monitor the pressure inside the suction pads. Proximity sensors and collision sensors detect obstacles around the robot. The robot can be moved forward and backward by making a sequence of predefined steps.

The block diagram of the embedded controller for the WCR is shown in Fig. 1. The embedded controller is based on Intel 80C188XL microcontroller with 64K EPROM and 64K RAM. The main function of the controller is to move the four legs of the robot with a predefined sequence during climbing operation and monitor its environment and react to it intelligently.

The embedded controller monitors its environment using some sensors. The environment must be monitored, typically, every half a second to detect the presence of obstacles using the collision and the proximity sensors during the forward and reverse movement of the robot. At each sampling period, the control signals to the DC motors are calculated using the proportional-derivative (PD) control algorithm. The current position of each leg joint is sensed using the position sensors and fed back to the embedded controller. The computation of the control signal typically, must be completed within 50 milliseconds to ensure the correct movement of the legs. The embedded controller also communicates with a remote PC to receive commands and sending back information via a serial communication link.

The main functional operation of the robot controller can roughly be divided into four major tasks: high-level navigation, environment monitoring, motor control and serial communication with remote PC. To satisfy the timing requirements for these major tasks, a real-time kernel is used to achieve multi-tasking in the control firmware.

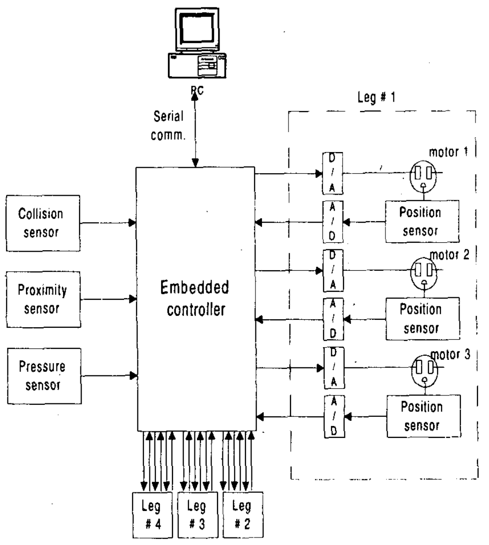

**Fig. 1**: Block diagram of the WCR controller.

## III. REAL-TIME SOFTWARE ENGINEERING METHODOLOGY

The purpose of software engineering methodology is to promote a certain approach to solve software problems [4]. Software engineering methodology provides guidelines in software development process and notation for analysis, design and documentation of the software developed. There are many software engineering methodologies developed over the years such as Structured Analysis and Structured Design (SA/SD), Jackson Structured Design (JSD), Object-Oriented Design (OOD) and others. However, most of these methodologies do not directly accommodate the development of real-time software since they do not consider non-functional requirements such as timing during the software engineering process [1]. As a result, new methodologies and extension of the existing methodologies were developed for the development of real-time software.

This section will review a structured RTSEM called Structured Analysis and Design for Real-Time Systems (RTSAD) and an object-oriented RTSEM called Unified Modeling Language for Real-Time (UML-RT). The reason for choosing RTSAD is because of its popularity in industry [5], [16] and UML-RT is currently among the most popular RTSEM in the market [2], [11].

### A. Structured Analysis and Design for Real-Time Systems (RTSAD)

RTSAD is an extension of Structured Analysis and Structured Design (SA/SD) to address the needs for real-time systems. There are two variations of RTSAD, the Ward-Mellor [15] and Hatley-Pirbhai [6] approaches. These two methodologies are extension of SA/SD in which behavioural characteristics of the system being developed are represented more precisely through the use of state transition diagrams, control flows and integrating state transition diagrams with data flow diagrams. In this paper only Ward-Mellor approach will be reviewed.

The Ward-Mellor method provides support for the analysis and design of software. Analysis is concerned with identifying the properties of a system, external events and how the external can influence the behaviour of the system, and then building a model called *essential model*, based on this information. There are two models developed in the analysis stage: *environmental model* and *behaviour model*. *Environmental model* defines the system in terms of its connections with the environments, while behaviour model describes the functions, dynamics and relationship in the system behaviour. Fig. 2 shows *context diagram* used to define *environment model* and Fig. 3 shows *data flow diagram* and Fig 4. shows *state transition diagram* used in *behaviour model* of the WCR control firmware.

Design stage is concerned with implementing the essential model on the hardware and software structures available to the designer. The implementation model composed of three models: the *processor environmental model*, the *software environmental model* and the *code organisation model*. The processor environmental model models the allocation of tasks to processors. The software environmental model used to examine the software aspect such as each processor requirements, examining the software architecture available of each processor and deciding on the software configuration. The *code organisation model* describes the modular structure of each task using structure chart. A *data dictionary* is created to capture all information from diagrams and charts used in the methodology.

## B. Unified Modeling Language for Real-Time (UML-RT)

UML-RT [3] is a third generation object-oriented modeling language. It adapts and extends the published works of Booch [7], Rumbaugh [17] and Jacobson [9]. UML-RT meant to be applicable to the modeling to all types of systems, it applies equally well to real-time systems and other kinds of "standard" software applications.

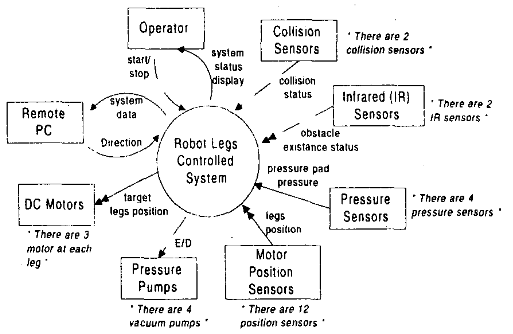

**Fig. 2**: WCR context diagram.

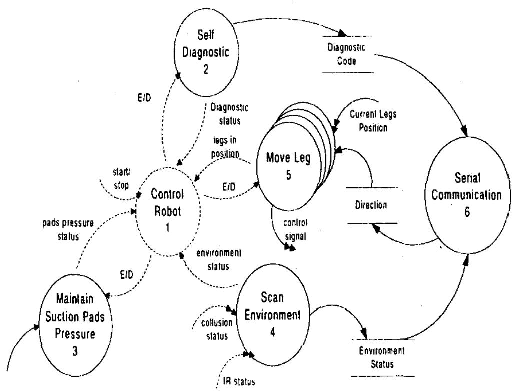

**Fig. 3**: Data and control flow diagram for WCR firmware.

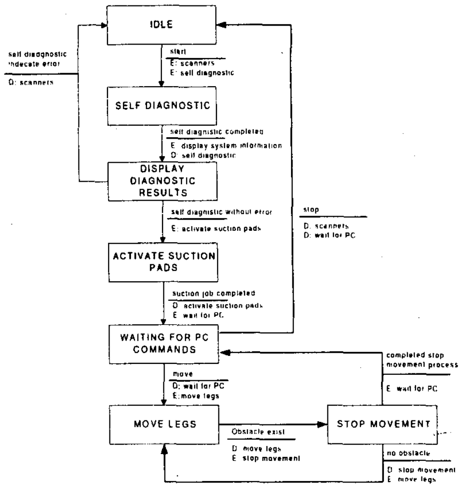

**Fig. 4**: WCR firmware state transition diagram.

Like Ward-Mellor, UML-RT also combines graphical and textual notations in order to model the systems. Modeling process using UML-RT consists of two major phases: analysis phase and design phase.

Analysis phases looks at key concepts and structures in the system that is independent of how the solution is implemented. There are three sub-phases in this phase: *requirement analysis*, *structure analysis* and *behaviour analysis*. *Requirement analysis* aims to identify and analyse the set of external object of significance and their interactions with the system. *Structure analysis* phase is to identify the key objects and classes and their relationship within the system itself. *Behaviour analysis* define and refine operations and behaviour (reactive behaviour and inter-object behaviour) of the system. Six types of diagrams are used in analysis phase are:

1) *External event object* diagram. It shows interaction between the system and external objects.
2) *Use case diagram*. They capture a broad view of the primary functionality of the system and how the system will be used, as shown in Fig. 5.
3) *Sequence diagram*. The diagram shows the sequence of messages between objects, as shown in Fig. 6.
4) *Collaboration diagram*. It shows the same basic information as *the sequences diagram* but *collaboration diagram* focus on the messages exchanged by a set of objects, refer to Fig. 7.
5) *Class diagram*. The diagram captures the object structure of the system, including the classes, objects, and their relationships.
6) *Statechart*. It shows the sequence of states for a reactive or interactive during its life in response to stimuli, together with its responses and actions. Fig. 8 shows the WCR environment monitoring statechart.

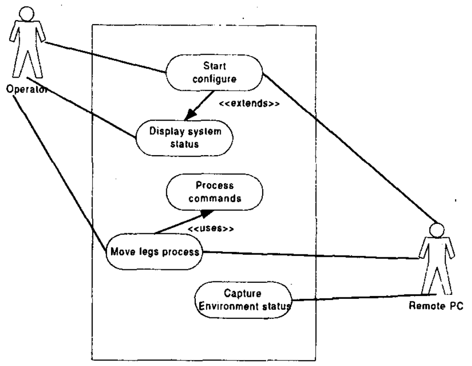

**Fig. 5**: WCR use case diagram.

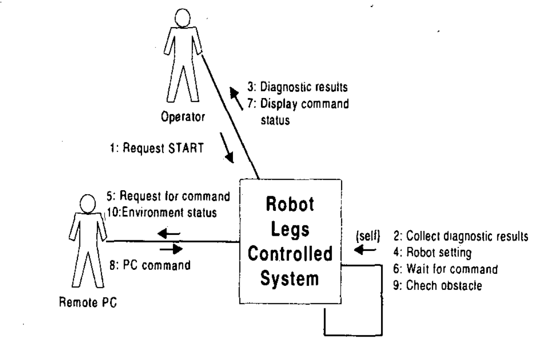

**Fig. 7**: Collaboration diagram for WCR starting system.

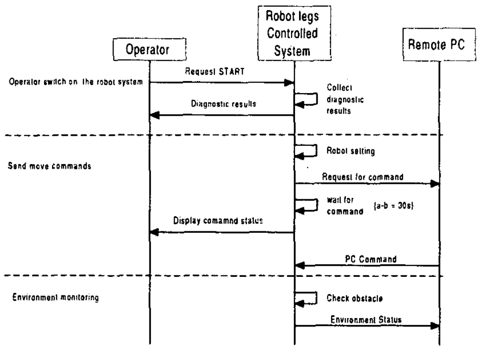

**Fig. 6**: Sequence Diagram for WCR starting system

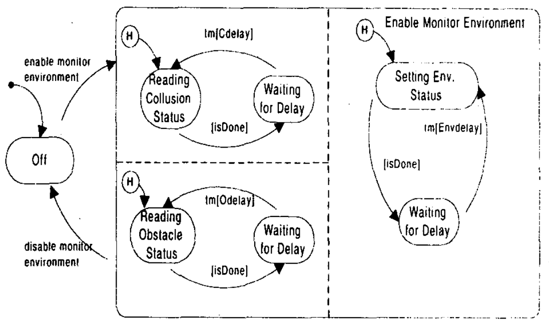

**Fig. 8**: Statechart for environment or sensors monitoring.

Design specifies a particular solution that is based on the analysis model. Design phase consists of *architectural design* phase, *mechanistic design* phase and *detail design* phase. *Architectural design* identifies the key strategies for the high level organisation of the system under development. *Mechanistic design* is the middle level of design, which deals with small set of classes, and objects collaborate to archive common goals. *Detail design* phase specify the detail internal structure such as storage format used for attributes, how associations are implemented, set of operation the object provides, selection of internal algorithms and specification of exception handling within the object. The UML-RT does define a special kind of state diagram, called an *activity diagram*, to decompose activities in the detail design phase.

## IV. COMPARISON OF TWO METHODOLOGIES FOR WCR CONTROL FIRMWARE

Both methodologies studied in Section III, capable of specifying functional and behavioural requirements in a real-time system. Specifically they provide models, methods and tools. UML-RT provides more diagramming tools for modeling than the Ward-Mellor method.

The criteria used for comparing the two methodologies are: software life cycle coverage, capabilities of methodology to support task structuring and concurrency model, timing analysis, maintainability and simplicity with good documentation. The developmental nature of the UTM WCR will also be considered in the comparison.

### A. Software Life Cycle Coverage

A methodology must support the entire software life-cycle which includes requirements analysis, requirements allocation, system and sub-system specification, preliminary design, detailed design and product specification. In these aspects both methodologies support the entire system development life-cycle.

As the WCR is developmental, the control firmware will evolve in functionality and complexity. When more functional specification is added, for example extending sensing capability to the robot, the methodology should be able to cope with these incremental changes. Any of the methodologies is sufficient for software development and they will cope with the incremental changes in specification. However, both methodologies do not provide sufficient guidelines on how to do the iterative and incremental development.

### B. Task Structuring and Concurrency Facilities

In the development of the robot control firmware a real-time kernel will be used to provide multitasking and concurrency facilities. Software design with a real-time kernel will be much easier if tasks are properly structured. Any methodology which can provides task structuring in the design stage will be beneficial to the development of the robot control firmware.

The Ward-Mellor methodology proposes a functional-based task structuring while UML-RT proposed object-based structuring. Object tasks structuring seem to be natural for the WCR controller, with sensor and actuator objects, and control objects between them.

The Ward-Mellor methodology addresses structuring the system into concurrent tasks, but it does not address the issues of task structuring in detail. UML-RT supports concurrency models in a couple of way using class diagram, statechart, collaboration and sequence diagrams. It seems that UML-RT support concurrency and task structuring in a clear and visible manner, but this have to be further investigated and evaluated.

However, none of the methodologies provide facilities to prioritise the structured tasks or objects. The priority allocation facility is important in development of WCR firmware because it will be developed using a pre-emptive real-time kernel where task priority will be allocated to tasks.

### C. Timing Analysis

The WCR control firmware is a categorised as hard real-time systems. In hard real-time system the issues of timeliness and dependability is the most critical issues, therefore methodologies which provide timing analysis will be very beneficial in the WCR software development.

Both methodologies address the timing analysis, but at different levels of detail in the timing analysis. Ward-Mellor addresses timing constraints during the analysis and design phase. The response time specification is developed during analysis phase and the timing requirements of each task including frequency of task activation and context switching overhead are determined during design phase.

UML-RT used sequence diagram and statechart to analyse the system timing by presenting the massages sequence, time constraints for particular massage patterns between objects and time constraints for particular action. Sequence diagram and statechart are used at all stages of analysis and design model. It aimed to represent the timing constraints at different level and different objects. To assist the representation of sequence diagram, collaboration diagram is used to provide message sequence numbering for all objects in UML-RT.

UML-RT can present timing constraint more visual and more detail using sequence diagram, statechart and collaboration diagram as compare to Ward-Mellor, which using state diagram.

### D. Maintainability

The evolving nature of the WCR system requires the firmware to be easy to maintain and modify. Traceability and adaptability of the WCR specification and design is important in order to make the firmware easy to maintain and modify. The methodology should be traceable in term of technique used in derivation of requirement and design.

In requirement specification stage, both methodologies describe the boundaries and the communication between the environment and the system. However, the environment model used in the methodologies differs in the aspects of scope of the system and the external events, and information presented.

In the Ward-Mellor's context diagram, devices used to interact with environment such as sensors and actuators are considered external events, whereas the context diagrams in UML-RT consists of object diagram along with appropriate stereotypes on messages and objects. This makes environment model for UML-RT less detail and there is a clear boundary between requirements and design. Consequently, UML-RT avoids the tendency to make design decision during the specification phase particularly if the specification gets detailed. This will be an advantage to the changing specification of the developmental WCR firmware.

### E. Simplicity and Good Documentation

The notation used in Ward-Mellor is much simpler and some of the constraints are represented in textual notation. The advantage is that at can be easily understood by users and minimum training is needed. This is important in the development of UTM WCR controller as the software designer are usually electrical engineers with little training in software development methodology. Because of the simple notation used, the non-functional requirements cannot fully be represented diagrammatically.

Ward-Mellor methodology was introduced almost twenty years ago, so it is stable and many references [14], [15] documentation [13] and reviews [10], [8], [12] of this methodology are available. Because of its simplicity and stability, the methodology is widely used in industry.

UML-RT is a methodology which rich of notation. A proper training is needed, in order to fully use the capability of the methodology. The use of CASE tools in this methodology is a necessity, because of the complexity of the notation used in the methodology. UML-RT is a relatively new modeling technique used in industry and academic research works, consequently there are few deficiencies and weaknesses in the methodology for some real-time applications. However modification and enhancement have been proposed by different companies and academic institution [2], [11].

## V. CONCLUSION

A proper real-time firmware methodology in needed in development of the WCR controller software in order to ensure the correctness and the quality of the software. Therefore, the suitability of real-time software methodologies for the WCR software development must be evaluated.

Both methodologies reviewed can support real-time concept and each methodology has their own specialty in solving real-time problems. Ward-Mellor is very popular in industry because of its simplicity to represent real-time problem and UML-RT is been currently promoted because it can provide almost a complete notation to represent soft real-time system. However, UML-RT does not yet have enough capability to represent hard real-time system.

## ACKNOWLEDGEMENT

The authors would like to thank UTM Mobile Robot Research Group (MRRG) for their cooperation in defining the wall-climbing robot requirement specification

## REFERENCES

[1] A. Burns, A. J. Wellings, HRT-HOOD: A Structured Design Method for Hard Real-time System, Volume 3, London: Elsevier, 1995.

[2] A. S. Events and A. J. Wellings, "UML and the Formal Development of Safety Critical Real-time Systems", IEE Applicable Modeling Verification and Testing, pp. 2/1-2/4, 1999.

[3] B. P. Douglass, Real-time UML Developing Efficient Object for Embedded Systems, Massachusetts: Addison Wesley, 1998.

[4] C. Ghenzzi, J. Mehdi and M. Dino, Fundamentals of Software Engineering, London: Prentice Hall, Singapore, 43, 1997.

[5] D. Rowe, "Current State of Local Industry Real-time Software Engineering Practice", Proceeding of the Center for Object Technology Advances and Research workshop (COTAR'95), pp. 73-81, Oct 1995.

[6] D.J. Hatley, and I.A. Pirbhai, Strategies for Real-Time System Specification, New York: Dorest House Publishing Co, 1987.

[7] G. Booch, Object-oriented Analysis and Design with Applications, US: Benjamin/Cummins, 1994.

[8] H. Gomaa, "Design Methods for Concurrent and Real-time Systems", Software Engineering, IEEE Computer Society Press, California, 1997.

[9] I. Jacobson et al., Object-Oriented Software Engineering, US: Addison-Wesley, 1992.

[10] L. Carroll, B. Tondu, C. Baron and J. C. Geffroy, "Comparison of Two Significant Development Methods Applied to the Design of Real-time Robot Controllers", IEEE International Conference on Systems, Man and Cybernetics, California, USA, Oct 1998.

[11] M. J. McLaughlin and A. Moore, Real-time extension to UML-timing, Concurrency and Hardware Interface, Dr. Dobb's Journal Dec. 1998.

[12] P. F. Tippell, "An Illustration of The Yourdon method with Ward/Mellor and Hatley/Parbhai Real-Time Extensions and a Review of Workstation Hosed CASE Tools That Support The Methodology", IEEE Colloquium on Computer Aided Software Engineering Tools for Real-Time Control, pp. 1/1 – 1/12, 1991.

[13] P. J. Pulli and M. Salmela, "Hard Real-Time Prototyping of SA/RT Specifications", EUROMICRO'91 Workshop on Real-Time, pp. 80-87, 1991.

[14] P. T. Ward, "The Transformation Schema: An Extension of Data Flow Diagram to Represent Control and Timing", IEEE Transactions on Software Engineering, Vol. 12, No. 2, pp. 198-210, February 1986.

[15] P. T. Ward, and S. J. Mellor, Structured Development for Real-Time Systems, Volume 1-3, New York: Yourdon Press, 1985.

[16] R. Whetton, M. Jones and D. Murray, "The Use of Ward and Mellor Structured Methodology for the Design of Complex Real-Time System", IEEE Colloquium on Computer Aided Software Engineering Tools for Real-Time Control, pp. 5/1 – 5/4, 1991.

[17] Y. Rumbaugh, M. Blaha, F. Eddy, W. Premerlani and W. Lorensen, Object-Oriented and Design, Prentice Hall, 1991.