Int. J. Advance Soft Compu. Appl, Vol. 5, No. 3, December 2013
ISSN 2074-8523; Copyright © SCRG Publication, 2013

A Weakly Hard Real-Time Scheduling Analysis Framework Using Genetic Algorithm

Habibah Ismail[1] and Dayang N. A. Jawawi[2]

Software Engineering Department, Faculty of Computing,
Universiti Teknologi Malaysia (UTM), 81310 Skudai, Johor, Malaysia
e-mail: [1]habibahisma@gmail.com, [2]dayang@utm.my

Abstract

In relation to real-time systems, hard real-time and soft real-time systems are based on "miss restriction" and "miss tolerance", respectively. However, a weakly hard real-time system integrates both these requirements. The problem with these systems is the limitation of the scheduling analysis method which only uses the traditional scheduling approach. Besides that, the current framework has problems with the complexity and predictability of the systems. This paper proposes a scheduling analysis framework based on, namely: the suitability of scheduling algorithms, the weakly hard real-time modelling and genetic algorithm approach for predicting the weakly hard real-time tasks. Initially, the best fitting specification of a weakly hard real-time system was integrated into the proposed framework and tested in the Modeling and Analysis of Real-Time Embedded systems (MARTE) profile. Sequence diagram complexity factor metrics were used to measure the behavioural complexity of the UML Profile for Schedulability, Performance and Time (SPT) and MARTE profiles. Further, the genetic algorithm approach was applied on the framework to measure the predictability of the soft real-time tasks. The results of the framework showed that the number of deadlines missed among the tasks was optimized by applying the framework on a case study. In conclusion, the proposed scheduling analysis framework provides a less complex design through the UML design modelling, as well as increasing the predictability of the systems.

Keywords: *Weakly hard real-time systems, scheduling analysis framework, UML modeling language, scheduling analysis, genetic algorithm approach.*

1 Introduction

Traditional real-time systems are classified into two categories, namely: hard real-time systems and soft real-time systems [1]. The new generation for a real-time system is the weakly hard real-time system. A hard real-time system is very restrictive because all the tasks must meet the deadlines or, in other words, no deadlines are allowed to be missed. Meanwhile, a soft real-time system is too relaxed as no guarantee can be given to the deadline, as to whether it is met or missed. As hard real-time and soft real-time systems are based on "miss restriction" and "miss tolerance" respectively, the weakly hard real-time system can integrate both of these requirements in which the distribution of its met and missed deadlines during a window of time is precisely bounded. For weakly hard real-time tasks, the missed or lost deadline happens occasionally and can be considered. However, it is still necessary and crucial to finish the tasks within a given deadline otherwise the tasks can result in failure. In a weakly hard real-time system, the number of deadlines that may be missed can be specified. Accordingly, this makes a weakly hard real-time system stronger than a soft real-time system.

A suitable framework for the schedulability analysis of real-time tasks can determine whether a specific task set derived from a software model can satisfy certain timing constraints and hence can be successfully scheduled. That framework enables scheduling analysis to predict the behaviour of critical tasks by meeting the deadline and, at the same time, predict the bounded way in which the missing of some deadlines is acceptable in comparison to less critical tasks [2]. A more realistic framework is required for the scheduling analysis of weakly hard real-time tasks because the constraints of missing deadlines do not exist in hard real-time task analysis frameworks and hence are not stated precisely in soft real-time task analysis frameworks [3]. Additionally, by using soft computing techniques such as genetic algorithm method in the prediction of the weakly hard real-time systems, the slight deadline missed can be predicted. This can be achieved using a fitness function from the precise soft real-time tasks specifications.

Modelling timing constraints and scheduling behaviour through the adaptation of modelling language is recognised as an alternative way by which to predict the timing behaviour and performance of set concurrent tasks in order to react to the changing environment [4]. This is due to the increasing complexity of contemporary ubiquitous real-time systems which require an adequate modelling language. The new extension for the Unified Modeling Language (UML) profile, called Modelling and Analysis of Real-Time Embedded (MARTE) system, has been standardised by the Object Management Group (OMG) to be the future standard for UML modelling of real-time and embedded systems, although a number of other modelling standards exist already [5]. This new profile is intended to replace the existing UML Profile for Schedulability, Performance and Time (SPT) since MARTE provides some new key features such as support for

non-functional property modeling. Further, it adds rich time and resource models to the UML.

This paper aims to contribute towards improved real-time scheduling by providing a scheduling analysis framework designed to predict the weakly hard real-time tasks. In order to cope with the increasing complexity in real-time systems, a modelling language is used. Further, the genetic algorithm approach is used for task scheduling in order to increase the predictability of weakly hard tasks in terms of the number of deadlines missed.

2 Related Works

For complex systems, besides using the scheduling algorithms only to schedule tasks and determine whether a task is schedulable or not, the algorithms can be used together with UML since this is a commonly accepted modelling language for complex systems [4]. The modelling profile must be able to cope with the complexity of the system, including the structure and behavioural aspects. As a result, it is essential to evaluate which model can cope with the complex structure and behaviour as well as with its non-functional requirements.

The behaviour of the system is known as being a set of external and internal sequences of events, actions and transitions [6]. It also can be said that the behaviour of a system is the response to the external events and the execution of actions that are taken at any time [7]. Hence, it is important to measure the behavioural complexity of design in weakly hard real-time systems in order to reduce the system's complexity.

Meeting all the deadlines is impossible; thus, [2] provided a conceptual framework for specifying real-time systems that can tolerate occasional losses of deadlines in which the distribution of the met and lost deadlines is precisely bounded. However, the existing framework is not yet sufficient to design predictable and correct weakly hard real-time systems; it requires some improvement.

Recently, scheduling algorithms with genetic algorithm (GA) are reported as in studies by [8][9][10]. Mitra and Ramanathan proposed a GA for the scheduling of non-pre-emptive tasks with precedence and deadline constraints [11]. Also, Monnier et al. presented a GA implementation to solve a real-time non-pre-emptive task scheduling problem [12]. These algorithms have been shown to have good results in real-time systems. The motivation is to integrate the GA into our framework because this algorithm can be used as an alternative solution to schedule. In addition, it can predict soft real-time tasks in which any slight violation of a deadline could be admitted.

3 The Proposed Scheduling Analysis Framework

This paper aims to make a contribution to the development of a framework for scheduling weakly hard real-time systems. Several works have appeared presenting different scheduling analysis frameworks. Among them are [2][3][13][14]. We have applied the scheduling analysis framework of [3] to express our framework since the approach/way used is easy to understand and appears to be the most suitable framework and the closest to our work. Moreover, their framework introduced steps as a guideline to express the framework. By using these steps, we can define the flow of the framework from the first step until the last step.

The proposed framework differs from the original, as portrayed in Fig. 1, to make it more understandable. We present a scheduling analysis framework that is flexible, less complex and increases the predictability as depicted in Fig. 1. The match steps in the framework are described in more detail as follows:

i. A task can be modelled using UML by illustrating the UML-SPT and MARTE sequence diagrams using the Rhapsody design modeling tool.

ii. The scheduling discipline is based on guaranteeing the satisfaction of the weakly hard constraints. Thus, offline schedulability checks or tests are required and scheduled under a fixed priority discipline.

iii. The worst case response time, denoted by R_i, of a task invocation is the $R_i \leq D_i$, which is a solution to the fixed point equation:

$$Ri = Ci + \sum_{\tau j \in hp(\tau i)} \left\lceil \frac{Ri}{Ti} \right\rceil Cj \tag{1}$$

where $hp(\tau_i)$ denotes the set of tasks having higher priority than task τ_i. C_i and C_j are the ith and jth task's worst case computation time. T_i is a constant, called the period of the task. If $R_i \geq D_i$, the weakly hard real-time tasks are scheduled according to the weakly hard analysis.

iv. The values of the hyperperiod are obtained by using the least common multiple tool. The equation for the calculation of the hyperperiod h_i is:

$$h_i = lcm\{T_j \mid \in hep(\tau_i)\} \tag{2}$$

where *lcm* is the least common multiple of the periods of the tasks and $hep(\tau_i)$ is the set of tasks of a priority higher than or equal to task τ_i.

v. The worst case interference of a task, τ_i with weakly hard constraints, λ_i and then the μ-pattern, denoted by $\mu_i(k)$, is given by:

$$\mu_i(k) = \begin{cases} 1 \text{ if } R_i(k) \leq D_i \\ 0 \text{ otherwise} \end{cases} \qquad k \geq 1 \tag{3}$$

vi. Whenever a task finishes (or at the deadline if the task has not been finished by the deadline) the μ-pattern is updated accordingly.

vii. When weakly hard tasks are scheduled, the basic weakly hard real-time modelling under UML modelling language (MARTE profile with some modification of their stereotypes and tags) is used in order to support the weakly hard timing requirements.

viii. When a task is finished with the scheduling analysis, the genetic algorithm is defined. A chromosome represents the relation of tasks and processors. The length of a chromosome L can be calculated as follows:

$$L = \sum_{i=1}^{N} n_i \tag{4}$$

ix. Group tasks with the same processor number and then calculate the deadline missing time and stop.

x. Compute the fitness function of individual (task), if deadline missed. The used fitness function, $F(s)$ has the following expression:

$$F(s) = 1 - \textit{success ratio} / D_i \tag{5}$$

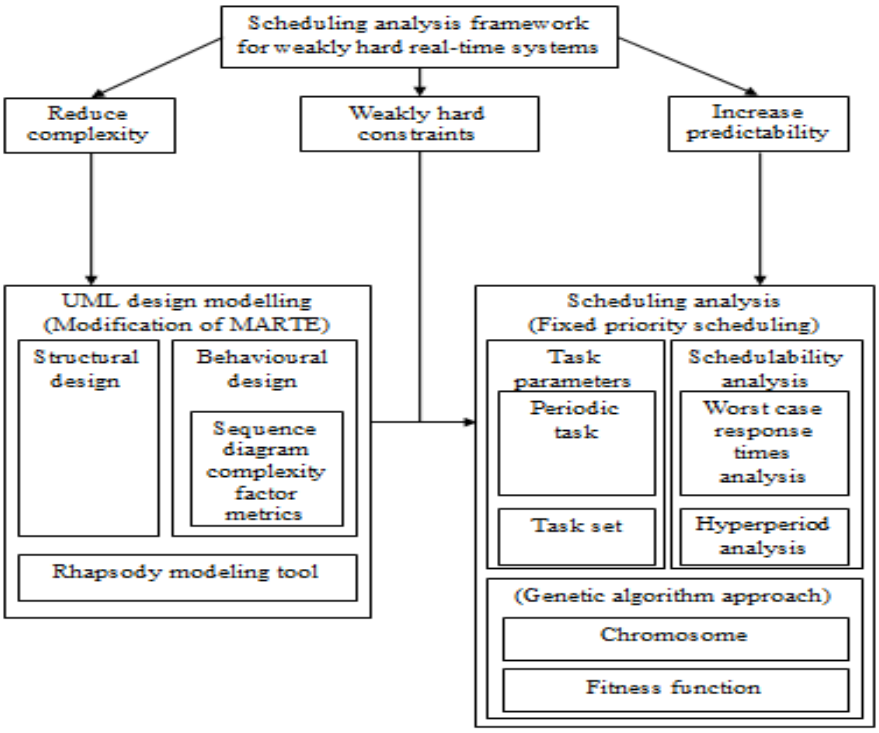

Fig. 1: Proposed scheduling analysis framework

4 An Autonomous Mobile Robot (AMR) Case Study

We chose the AMR case study introduced by [15] because it consists of hard tasks and soft tasks. Thus, it is unnecessary for the system to meet the entire task and message deadlines as long as the misses (or deadlines) are spaced distantly/evenly or, in other words, are composed of weakly hard tasks. The AMR system we used has more than one processor called a multiprocessor and processes the tasks on different processors.

We first present a task set as an example by which to illustrate the distribution of met and missed deadlines under fixed priority scheduling. There are seven tasks derived from the AMR system, and all the tasks are listed in Table 2. The AMR system used in the case study is a differential drive wheeled mobile robot, capable of traversing in a structured environment. The goal of the robot software is to control the movement of the robot while avoiding obstacles in its environment.

The structure of this section is as follows and is related with our proposed scheduling framework: Sub-section 4.1 presents the UML design modelling and discusses Step (i). It consists of sub Sub-section 4.1.1 which discusses the behavioural complexity of UML-SPT and MARTE sequence diagram. The scheduling analysis is placed in Sub-section 4.2 and explains Steps (ii) to (vi) in detail. In Sub-section 4.3, there is a discussion about Step (vii). Following that,

Sub-section 4.4 presents the genetic algorithm approach and then, Step (viii) until (x) is discussed.

4.1 UML design modelling

We started the mobile robot case study in relation to the development of real-time software systems using UML-SPT and MARTE profiles. Fig. 2 illustrates the UML-SPT and MARTE sequence diagram in schedulability analysis modelling. The UML-SPT sequence diagram in performance analysis modelling is illustrated in Fig. 3. Both of the sequence diagrams were created using the Rhapsody modeling tool.

Generally, a sequence diagram is commonly used to show the behavioural aspects of the system since, in UML, the behavioural aspects of the system are best described by using the sequence diagram [16]. A sequence diagram can be also easily understood. A sequence diagram consists of some basic elements including frames, regions, objects' roles, lifelines, arrows (for showing the message type i.e., synchronous or asynchronous), messages and calls (for showing operations). Upon review and study of both profiles, it can be seen that there are similarities in the structural aspects because there is no difference in the class diagram used. Therefore, we have made a comparison regarding the behavioural aspect in view of the behavioural complexity of the sequence diagram for both profiles. The steps needed to calculate the behavioural complexity and the relevant equations are discussed in the following sub-sections.

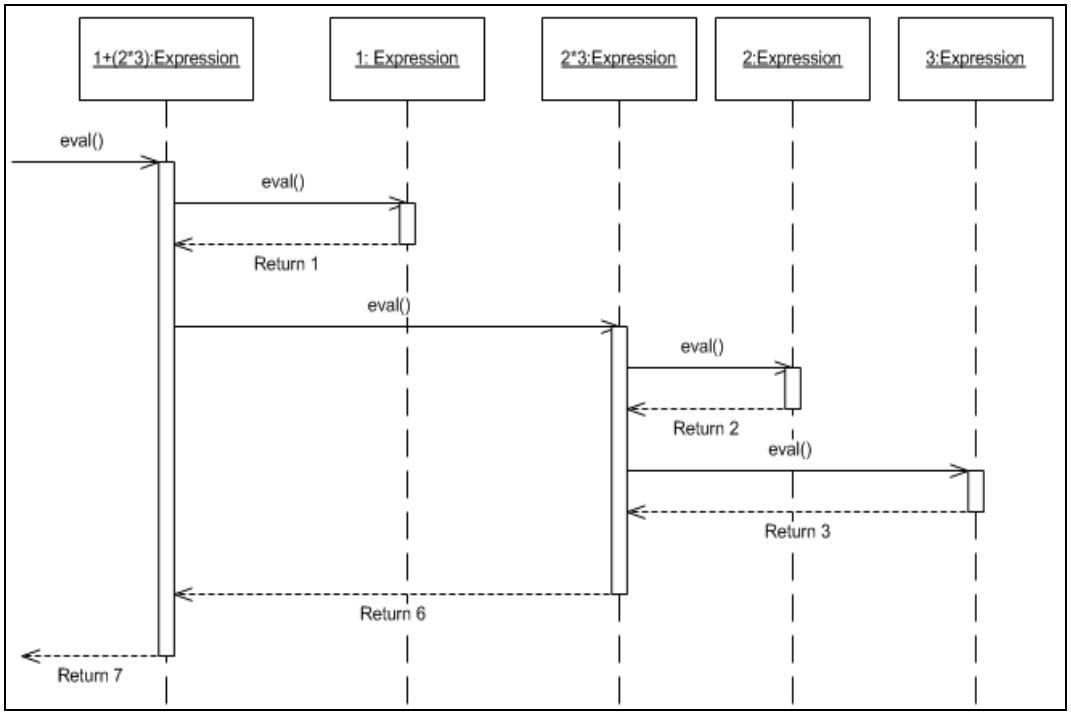

Fig. 2: UML-SPT and MARTE sequence diagram in schedulability analysis modelling using Rhapsody tool

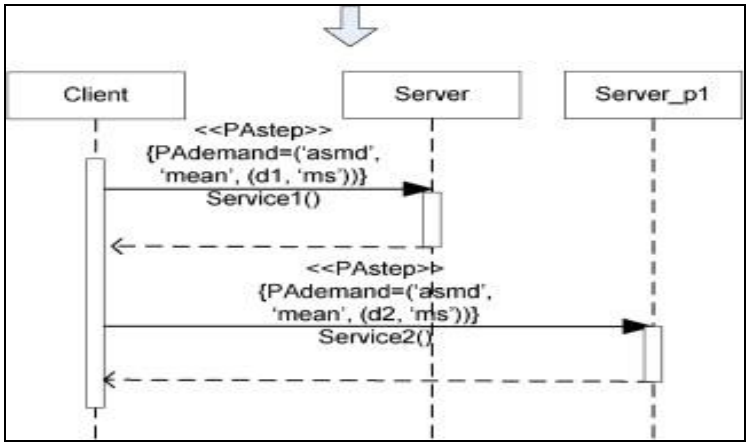

Fig. 3: UML-SPT sequence diagram in performance analysis modelling

4.1.1 Behavioural complexity of UML-SPT and MARTE sequence diagram

The behavioural complexity of the UML-SPT and MARTE sequence diagrams is measured by using the Sequence Diagram Complexity Factors (SDCF) metric, as introduced in [17]. In this sub-section, the sequence diagrams are compared based on the results produced by the metric. The same methods and attributes were used

in the UML-SPT and MARTE profiles while the mobile robot system was being designed.

The basic elements that compose a sequence diagram are objects and the messages between them. This metric signifies the complexity of a sequence diagram due to the objects used in them:

$$SOW_{adj} = \sum (\text{No. of objects} * \text{object type factor})$$

This metric indicates the complexity of a sequence diagram due to the messages used in them:

$$(SMW)_{un} = \sum (\text{No. of messages} * \text{message type factor})$$

Considering the equation of SMW unadjusted (SMW_{un}), the sequence method utilisation factor (SMUF) as the total number of objects from which a method is getting initiated, is defined. Thus, SMUF = $\sum$ (No. of objects of the same sequence diagram) is used. Therefore, the adjusted value of SMW is:

$$SMW_{adj} = SMW_{un} + SMUF$$

The overall complexity of a sequence diagram (C(SQ)) will be the combined value of Sequence Object Weight (SOW) and Sequence Message Weight (SMW). Therefore, the complexity of a sequence diagram is:

$$C(_{SQ})_{un} = SOW_{adj} + SMW_{adj}$$

Table 1: Results of behavioural complexity of designed sequence diagrams

UML-Profile / **Variable**	**UML-SPT**	**MARTE**
SOW_{adj}	60	30
SMW_{un}	84	42
SMUF	20	10
SMW_{adj}	104	52
C(SQ) AutonomousMobileRobot	**164**	**82**

The results for each parameter are presented in Table 1. It is noted that the SOW adjusted for MARTE is less than for the UML-SPT, as well as for the unadjusted SMW. The result for the total number of objects or Sequence Method Utilization Factor (SMUF) (this equation is for the adjusted SMW) still shows MARTE as being less than UML-SPT. The result for the adjusted SMW is

obtained by adding the unadjusted SMW with SMUF and the result shows MARTE is less than UML-SPT. Finally, we secure the result for the complexity of a sequence diagram denoted by C(SQ) wherein MARTE is shown to be less complex than UML-SPT. Hence, from the overall results, we can conclude that the behavioural complexity of MARTE is less than that of UML-SPT. In addition, it can be seen that MARTE can reduce the behavioural complexity as the values of complexity obtained were small.

As clearly shown in Table 1, the design of a weakly hard real-time system by using the UML-SPT sequence diagram design is much more complex as compared to the MARTE sequence diagram. MARTE provides an incredible reduction of the behavioural design complexity in the sequence diagram. Based on the SDCF metric measurement, MARTE decreases the complexity of the sequence diagram compared with the UML-SPT model. As the results in Table 1 show, the behavioural complexity of the mobile robot sequence diagram in MARTE was almost half that in UML-SPT. One reason for this is that the UML-SPT needed two sequence diagrams in order to model schedulability and performance data analysis, while only one sequence diagram was sufficient in MARTE modelling. In addition, we conclude that the complexity of a diagram does not depend on the design tools used, but rather on how many tasks there are in the diagram for each profile.

We can conclude that the MARTE sequence diagram is the best profile in order to reduce the complexity of system behaviour compared to the UML-SPT based on the results of behavioural complexity of designed sequence diagrams as depicted in Table 1. Thus, the MARTE profile appears to be the most suitable profile in terms of less complex design.

4.2 Scheduling analysis (Fixed priority scheduling)

Meeting timing or deadline constraints is one of the most important concerns in real-time systems. In the parameters presented in the Table 2, T_i is a constant, called the period of the task, and D_i refers to the deadline of the task that must be completed. Every task is periodic and we assumed that $D_i = T_i$. C_i represents the worst case execution time of each cycle of the tasks, and the weakly hard temporal constraints, λ and R_i, represent the worst case response time of the tasks assuming fixed priority scheduling. The higher order period or the hyperperiod, h_i, consists of the number of invocations of a task in the hyperperiod at level i, $a_i = \frac{h_i}{Ti}$.

From Table 2, even though *Cruise* and *manrobotintf* tasks missed their deadlines in the worst case scenario, by using hyperperiod analysis, we can specify the number of deadlines missed for both tasks. The *Cruise* task was invoked $\alpha_6 = 4$ times within the hyperperiod at level 6 while the *manrobotintf* task was invoked $\alpha_7 = 8$ at level 7. Therefore, in the worst case scenario, for the *Cruise*

task, only two invocations out of four would miss the deadline. Thus, the weakly hard constraint for the *Cruise* task is defined as $\binom{2}{4}$ constraint. Meanwhile, for the *manrobotintf* task, only one invocation out of three would miss the deadline, which means the weakly hard constraint for the *manrobotintf* task is defined as $\binom{1}{3}$ constraint.

The main objective of weakly hard constraints is to guarantee that the tasks meet their timing constraints, even though some deadlines may be missed during execution time. Meanwhile, μ-patterns are used to determine how the deadline can be missed in terms of the consecutiveness or non-consecutiveness of such missed and met deadlines.

Table 2: Task parameters of the task set

Task	T_i	D_i	C_i	R_i	h_i	a_i	λ	**μ-pattern**
MobileRobot	50	50	1	1	50	1	(1,1)	{1}
motorctrl_left	50	50	20	21	50	1	(1,1)	{1}
motorctrl_right	50	50	20	41	50	1	(1,1)	{1}
Subsumption	80	80	1	42	400	5	(5,5)	{11111}
Avoid	100	100	17	100	400	4	(5,5)	{11111}
Cruise	100	100	1	160	400	4	(2,4)	{1100}
manrobotintf	150	150	16	236	1200	8	(1,3)	{100}

From the results, it is clear that using the weakly hard constraints specification allowed us to model tasks like this one by specifying the bounds on the number of deadlines a task may miss, as well as on the pattern of how these deadlines can be missed by finding the hyperperiod for each task. Therefore, we can establish how many times each task is invoked within the hyperperiod and from the invocations, we can obtain the number of deadlines missed and met for each task.

Most importantly, the tasks were still guaranteed to be finished within a given deadline. There were, however, two tasks that missed their deadline in the worst case scenario because the task satisfied its temporal constraints. Indeed, missing some of the deadlines is acceptable in weakly hard real-time systems.

4.3 Modification of MARTE modeling

The modification of MARTE profile is necessary in order to overcome the problem with the profile. This problem resulted in its timing constraint being restricted to hard and soft real-time systems. Thus, in order for the profile to support weakly hard real-time requirements, the modifications are necessary. We modified the UML-MARTE metamodel by adding the weakly hard requirements into the <<*saEnd2EndFlow*>> stereotype. Accordingly, we added weakly hard specifications, namely *µ-pattern* and *weakly hard temporal constraints* as tags of that stereotype [18].

The modification results are applied in the AMR case study. To show how this is done, the AMR case study is depicted based on the chosen scenario. All the tasks in the AMR case study were as shown in Table 2 in Sub-section 4.2. The scenarios were then described to analyse how the tasks were defined in reaction to the external or internal events and how they executed the actions in response.

We used a common tool for UML modeling that is supported by MARTE, namely Rhapsody software modelling tool. Based on these scenarios, the schedulability analysis of the weakly hard tasks is measured. Fig. 4 presents the AMR sequence diagram used to show the timing requirements for the *Cruise* task in the MARTE profile.

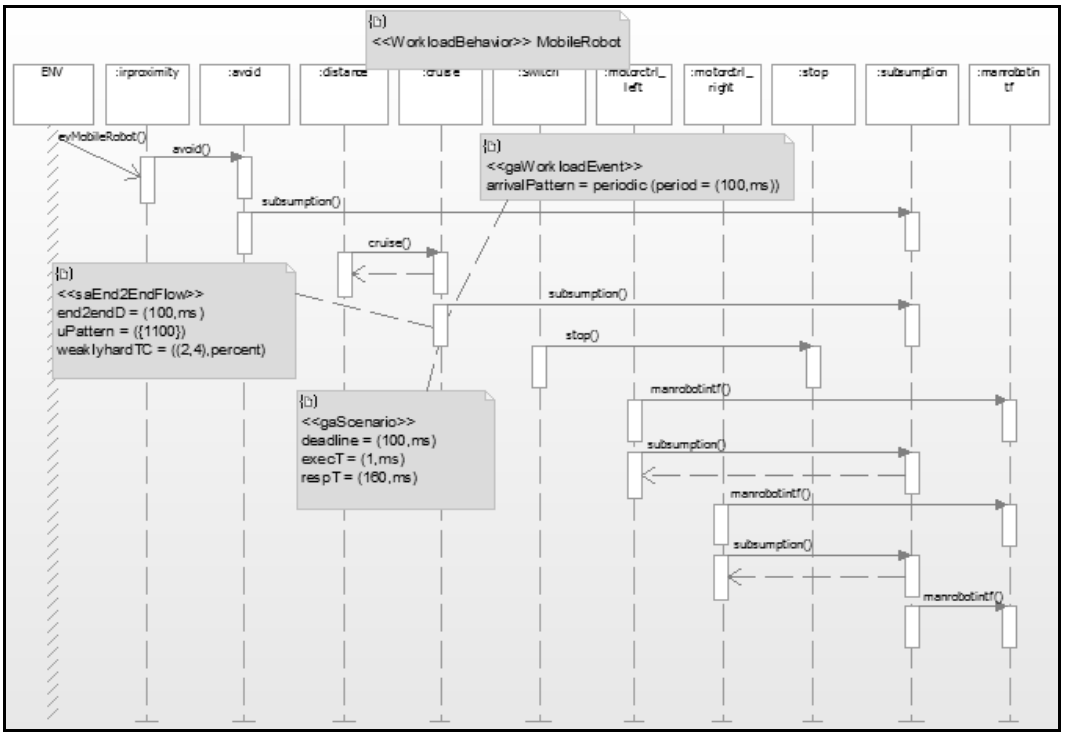

Fig. 4: MARTE sequence diagram of AMR case study

4.4 Scheduling analysis (Genetic algorithm approach)

Traditionally, the performance of a scheduling algorithm is measured by the utilization of a processor and the worst case response time of tasks. As the rate of real-time applications grows, the necessity of scheduling an algorithm for weakly hard real-time systems is increasing. In this sub-section, we have focused on the scheduling for weakly hard tasks using genetic algorithm. Genetic algorithm is a learning way that is based on biologic evolution. This algorithm is a technique for finding approximate solutions for optimization. The step we used during genetic algorithm is that of a chromosome structure, having requirements for number of processors, number of tasks and fitness functions.

Based on the tasks generated by AMR as well as those depicted in Table 2 in Sub-section 4.2, the following guidelines are produced to use genetic algorithm to predict the scheduling of tasks of real-time systems. The tasks must be divided into two categories of timing requirements as described below:

- Hard real-time tasks related to the motor control of the robot and robot activity, namely: MobileRobot, Avoid, motorctrl_left, motorctrl_right and Subsumption.
- Soft real-time tasks that are related to the robot activity and display task, namely Cruise and manrobotintf.

As stated, missed deadlines are not allowed for hard tasks or those five tasks and the tasks must be completed within their deadlines. Among the seven tasks, five tasks are known as hard tasks and two tasks are known as soft tasks. Thus, missing deadlines for soft tasks is acceptable for the AMR system. It is natural to represent a schedule of tasks in a chromosome wherein each task has a deadline time. After the encoding of chromosomes has occurred the fitness value of each task is then calculated. A typical schedule as a chromosome vector for a set of seven tasks, to be executed on a multiprocessor system with 2 processors, is shown in Table 3.

Each task T_i is characterized as follows: R_i is response time, C_i is worst case computation time and D_i is deadline. The scheduler determines the scheduled start time and finish time of each task. The scheduler works in parallel with the processors. If $st(T_i)$ is the scheduled start time and $ft(T_i)$ is the scheduled finish time of task T_i, then the task T_i, is said to meet its deadline if $(R_i \leq st(T_i) \leq D_i - C_i)$ and $(R_i + C_i \leq ft(T_i) \leq D_i)$. That is, the tasks are scheduled to start after they arrive and finish execution before their deadlines. A set of such tasks can be said to be guaranteed.

Clearly, Table 3 represents the structure of a chromosome by creating the scheduling set on each processor. A chromosome contains a value of computation time and the respective deadlines. In this table, seven tasks are assigned to two processors, with total scheduling time of 150. The length of chromosome is equal to the number of tasks, since all of the tasks must be executed. Hence, the length of chromosome L is 7. The scheduling task set on processor 1 P_1 is generated by abstracting tasks with the value of computation time, and C_i and P_2 are also generated in the same way. The tasks were arranged in the increasing order of their deadlines and randomly assigned to processors considering fixed priority scheduling based on the priority of the tasks. Tasks *T1*, *T2*, *T3*, *T4* and *T5* are assigned to processor 1, while tasks *T6* and *T7* are assigned to processor 2 respectively and scheduled for execution successfully. However, tasks *T6* and *T7* cannot be executed on processor 2 respectively and it is considered as a missing deadline.

Task *T7* cannot be scheduled until *T6* has been completed. Further, *T6* is predecessor of *T7*, and this *T7* is the successor of *T6*, under the relation of dependency on a multiprocessor system.

Table 3: The structure of chromosome

	T1	T2	T3	T4	T5	T6	T7
C_i	1	20	20	1	17	1	16
D_i	50	50	50	80	100	100	150
	P1	P1	P1	P1	P1	P2	P2

T_i = Task set; P_i = Processor;

Here, we consider some general schedules of task T_i, on the processor system for a multiprocessor. In the worst case scenario, the *T6* and *T7* tasks cannot be scheduled, thus resulting in a need for a genetic algorithm to find the number of missed deadlines for the tasks using the fitness function, sometimes called the objective or evaluation function. For the other tasks in which it can be scheduled, the fitness function value is equal to zero. The fitness function is essentially the objective function for the problem. A fitness function quantifies the scheduling chances for a given deadline set (chromosome). In Equation (5), $F(s)$ is a fitness function, while the success ratio = *number of scheduled tasks* divided *total number of tasks*, and D_i refers to the deadline.

$$F(6) = 1 - 0.71 / 100$$
$$= 0.003$$

$$F(7) = 1 - 0.71 / 150$$
$$= 0.002$$

The results show that even though the *T6* and *T7* tasks missed their deadlines, the system is weakly hard. As a result, it is schedulable because the probability of the deadline being missed is less than 1. Thus, we could still consider the system schedulable since we were willing to admit $\rho < 1$ (ρ being a parameter) of deadline misses. Also, the genetic algorithm can be used to schedule tasks so as to meet deadlines.

In order to generate an optimal schedule for the autonomous mobile robot, we can change the priority of tasks depending on its category, in which the hard real-time tasks should take the highest priority among all tasks. Because a slightly missing deadline in soft real-time tasks can be allowed depending on the actual robot behaviour, we found that the robot can work without any problem arising from missed deadlines for soft real-time tasks.

5 Conclusion

In conclusion, this paper has proposed scheduling analysis framework that can address the two main problems. The first problem was that of complexity. In order to reduce the complexity of the system, we used two well-known UML, namely the UML-SPT and UML-MARTE profiles, together with scheduling analysis. The system complexity was reduced by using weakly hard real-time specifications together with the UML-SPT and UML-MARTE profiles. In order to know which profile was less complex, we evaluated the behavioural complexity of both profiles using SDCF metrics and the results showed that the UML-MARTE profile was less complex compared with the UML-SPT profile.

The second problem was related to predictability. This was investigated by using the genetic algorithm approach. This approach brings significant benefits in which any deadline violation can be measured and the number of deadlines missed can be guaranteed; hence, it can increase the predictability of the tasks.

In this systematic way, designers can provide less complex and more predictable weakly hard real-time systems based on the modelling approach, scheduling approach and scheduling algorithm respectively. This is because the proposed scheduling analysis framework has the ability to reduce the complexity of real-time systems by using the UML and scheduling approach and also has the ability to increase the predictability of the tasks based on the genetic algorithm approach. For future work, we aim to evaluate the proposed scheduling analysis framework by comparison with the existing framework.

ACKNOWLEDGEMENTS

The authors would like to profoundly thank the Research Grant University (RUG), the Fundamental Research Grant Scheme (FRGS) from MOHE, ScienceFund and Ministry of Science, Technology and Innovation Malaysia (MOSTI) Vote No. 4S064, as well as the Universiti Teknologi Malaysia (UTM) for their financial support and, not forgotten, the Software Engineering Research Group (SERG) and EReTSEL lab members for their help.

References

[1] K.G. Shin and P. Ramanathan. 1994. Real-time computing: A new discipline of computer science and engineering, *Proceedings of the IEEE*, vol. 82(1).

[2] G. Bernat. 1998. Specification and analysis of weakly hard real-time systems, PhD Thesis, Department de les Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, January 1998, Spain.

[3] G. Bernat and A. Burns. 2001. Weakly hard real-time systems, *IEEE Transactions on Computers*, vol. 50(4), pp. 308-321, April 2001.

[4] K. E. A. Jensen. 2009. Schedulability analysis of embedded applications modelled using MARTE, Master's Thesis. Technical University of Denmark, Kongens Lyngby, Denmark.

[5] OMG MARTE Specification. 2007. A UML Profile for MARTE: Beta 1, OMG Adopted Specification ptc/07-08-04, August.

[6] D. Harel and E. Gery. 1997. Executable object modeling with statecharts, *Computer*, vol. 30(7), pp. 31-42.

[7] Rational Software Corporation. 2003. C++ Reference. Rational Rose Real-Time. Version: 2003.06.00, Part no: 800-026109-000, *Technical report of Rational Software Corporation*, ftp://ftp.software.ibm.com/software/rational/docs/v2003/unix_solutions/pdf/RoseRT/rosert_cpp_ref_guide.pdf.

[8] M. Yoo and M. Gen. 2005. Multimedia tasks scheduling using genetic algorithm, *Asia Pacific Management Review*, vol. 10(6), pp. 373-380.

[9] J. Oh and C. Wu. 2004. Genetic-algorithm-based real-time task scheduling with multiple goals, *Journal of Systems and Software*, vol. 71(3), pp. 245-258.

[10] G. Sebestyen, K. Pusztai and Z. Puklus. 2004. Genetic algorithm for real-time scheduling in distributed control systems, *International Conference on Intelligent Engineering* (INES'04), pp. 117-120.

[11] H. Mitra and P. Ramanathan. 1993. A genetic approach for scheduling non-pre-emptive tasks with precedence and deadline constraints, *Proceedings of the 26th Hawaii International Conference on Systems Sciences*, pp. 556-564.

[12] Y. Monnier, J. Beauvais, and A. M. Deplanche. 1998. A genetic algorithm for scheduling tasks in a real-time distributed system, *Proceedings of 24th Euromicro Conference*, pp. 708-714.

[13] A. Burns and G. Bernat. 2001. Jorvik: A framework for effective scheduling, *Real-Time Systems Research Group*, Department of Computer Science, University of York, UK.

[14] A. Burns, G. Bernat and I. Broster. 2003. A probabilistic framework for schedulability analysis, *Lecture Notes in Computer Science*, vol. 2855.

[15] D. N. A. Jawawi, S. Deris and R. Mamat. 2006. Enhancement of PECOS embedded real-time component model for autonomous mobile robot application, *IEEE International Conference on Computer Systems and Applications*, pp. 882-889.

[16] G. Booch, J. Rumbaugh and I. Jacobson. 2005. Unified Modeling Language User Guide, the (Addison-Wesley Object Technology Series). Addison-Wesley Professional.

[17] A. Kanjilal, S. Sengupta and S. Bhattacharya. 2009. Analysis of complexity of requirements: A metrics based approach, *In Proceedings of the 2nd India Software Engineering Conference* (ISEC'09), pp. 131-132.

[18] Habibah Ismail and Dayang N. A. Jawawi. 2011. The adoption of UML-MARTE profiles for weakly hard real-time requirements, *5th Malaysian Conference in Software Engineering* (MySEC'11), December 13-14.

[19] S. Agrawal, R. Shankar and Ranvijay. 2011. A three phase scheduling for system energy minimization of weakly hard real-time systems, *Global Journal of Computer Science and Technology*, vol. 11(10), version 1.0.