An Approach for Predictable Real-Time Software in Component-Based Software Development of Autonomous Mobile Robot Systems

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

ABSTRACT

The abilities to predict and analyze timing are key requirements for many real-time software such as the software for an Autonomous Mobile Robot (AMR) system. A self-contained AMR system requires on-board computation and the system is typically constrained by limited processing power and memories. However, the software needs to fulfill its timing and application requirements, despite the constraints. Thus, the capability of real-time theories to predict the AMR performance against the timing requirements is very significant. The aims of this paper are to proposed and demonstrate a detail approach to enable predictable of AMR real-time performance in the design phase of component-based software development. This high level prediction can avoid timing error in the field and costly late rework at the implementation phases. An experiment was designed, to demonstrate the approach and to validate the predicted results against the real performance of the AMR software.

KEYWORDS

Embedded real-time systems, component-based software development, rate-monotonic analysis and component model.

1. Introduction

Component-Based Software Development (CBSD) is gaining popular approach to increase software productivity, improve software quality, and at the same time decrease the costs of software development. CBSD has been widely used in applications such as office PC, distributed web-based and e-business. Unfortunately, in Embedded Real-Time (ERT) systems, it is difficult to directly adopt current component-based approach in the domain especially for small ERT systems due to the resource-constrained and real-time requirements of the systems [1]. A number of research and industries have successfully applied CDSD in ERT systems by optimizing their approach for speed and memory consumption during components composition at design time [2][3][4][5]. This enables global optimizations and prediction of the systems requirements from the given component properties.

To create a predictable ERT system, predictability with respect to timing requirements is important for reliable real-time systems because the correctness of the system depends not only on the logical results, but also on the time at which the results are produced. The real-time control of an Autonomous Mobile Robot (AMR) provides an example of a precise observation and prediction of timing constraints is a necessary condition for guaranteeing a stable behavior of the robot. 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. The theories developed on real-time computing such as scheduling analysis algorithms can be practically used in robotics application to make the robot systems more reliable [6].

Scheduling analysis is a fundamental tool for checking timing correctness of a real-time application. It allows checking timing constraints by predicting the worst-case behavior of a real-time system when a scheduling algorithm is applied. To perform schedulability analysis on a component-based software, the integration of the component model and the real-time scheduling theory is important. This integration process enables predictable assembly of components.

Previously, we proposed and evaluated a general approach for predictable real-time software in component-based development of AMR systems [7]. This approach is derived from Predictable Enabled Component Technology (PECT) developed by Carnegie Mellon Software Engineering Institute (SEI) [8]. The proposed approach consists of an analytical model based on Rate Monotonic Analysis (RMA) schedulability analysis [9] that is packaged together with a component technology called PErvasive COMPONENT Systems (PECOS) model [4].

PECT allows freedom for the developer to select any model to be used with their method. However, in our case, in which most of the AMR software developers are not from software engineering background and with little knowledge in real-time system theories, this high degree of freedom might prevent the possibility to achieve predictable capabilities in components assembly.

Here, we proposed a detail approach for predictable composition of AMR components based on PECT aims at limiting the freedom and achieving predictability. This approach enables predictable AMR real-time performance using a schedulability analysis in CBSD. The prediction made using the proposed approach is experimentally verified against a real-time implementation of an AMR system. In this experiment
the reliability and reactivity of the robot behavior are analyzed.

This paper is organized as follows. The proposed approach to predict real-time performance of an AMR system is described in the Section 2. Experimental results on the real-time implementation of the AMR and the prediction of the AMR performance using the proposed approach will be discussed in detail in Section 3. Finally, the discussion and conclusion of this paper are presented in Section 4.

2. Predictable Assembly for AMR Systems

The proposed detail approach consists of the following steps: 1) Capturing temporal models in PECOS; 2) Performing utilization feasibility analysis; and 3) Performing exact feasibility analysis. In this section, these steps are further described based on AMR software models. The component models and real-time theories used in developing each steps in the approach will also be introduced.

2.1 AMR Software Models

In modeling AMR software, there are at least three standard models which will influence the architecture of the software. Traditional AMR software model is based on the sense-plan-act organization have been criticized due to the emphasis placed on construction of a world model and planning actions based on this model. The computation time required to construct a symbolic model has a significant impact on the performance of the robot. An alternative control organization is to use behavior-based model which is a reactive system that do not use symbolic representation, and have been demonstrated capable of producing reasonably complex robot behavior. Hybrid model is the third model which usually modeled in three layers: reactive layer for real-time tasks; middle layer for supervisor tasks; and deliberate layer for planning, localization, reasoning and interaction with human operator [10].

Currently, we are focusing on reactive layer of hybrid model using behavior-based intelligent control approach [11], since, software at reactive layer is typically constrained by limited resources and real-time requirements. A simple AMR case-study is used in this section to illustrate the design of reactive layer AMR software using the behavior-based intelligent control approach.

The AMR 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 AMR consists of a body and a pair of wheels. Each drive wheel is move by a direct-current (DC) motor. The speeds of the motors are sensed using shaft encoders and fed back to the embedded controller for computation of control signal to the DC motor every 10 to 50 milliseconds (msec) using the proportional-integral (PI) control algorithm. The embedded controller also monitors the robot environment using four infrared (IR) proximity sensors and switches. Figure 1 shows the behavior layer architecture of the case study for an intelligent control of the AMR systems. The three behaviors provide reactive operation to the AMR in uncertain dynamic environment.

Figure 1: The AMR behavior layer architecture.

2.2 Temporal Model in PECOS

The major steps involve in modeling the AMR software using PECOS component model are identification of required components from the AMR application requirements, composition of components, and assigning the period and priority of components. The product from these steps is a component composition diagram. We enhanced the connection in PECOS model by allowing connection of constant values to input ports [12]. Figure 2 illustrates composition of seventeen components in enhanced PECOS model for the AMR system including the subcomponents in the three composite components. In the figure, composite components are shown by blocks with shadow.

In PECOS, components can be any of three types: active, passive or event component. Active components have their own thread of control; passive components do not have their own thread of control; and an event component is a component that is triggered by event. In our integration process only passive and active components are considered, since, each event component can be converted to an active component using poling technique. The composition shown in Figure 2 consists of eight active components and nine passive components (three passive components in each composite components).

In Figure 2, components with period and priority fields are active components and components without the fields are passive components. In this composition we added one component for operator monitoring purposes. The AMR can communicate with human through a Liquid Crystal Display (LCD) by displaying the robot status, behavior and current speeds. Based on static structure of Figure 2, it can be concluded that component composition using PECOS model can explicitly describes the temporal models and high level structure of the AMR software.
2.3 Utilization Feasibility Analysis

Lui and Layland developed the original Rate-Monotonic Analysis (RMA) in the 1970s [13]. It was developed starting from a very simple model where all tasks are periodic and independent. The simple RMA in Theorem 1 is for checking the schedulability of periodic and independent task set.

**Theorem 1**: A set of \( n \) independent periodic tasks is schedulable by rate monotonic algorithm, if

\[
\sum_{i=1}^{n} \frac{C_i}{T_i} \leq U_n = n(2^{\frac{1}{n}} - 1), \quad n = 1, 2, \ldots
\]

where \( C_i \) is the \( i \)th task’s computation time, \( T_i \) is the \( i \)th task’s period and \( U_n \) is the utilization bound for \( n \) task.

The worst case utilization bound in the theorem are assume to be tight in the sense that there exist some infeasible task sets with utilization arbitrarily close to \( n(2^{\frac{1}{n}} - 1) \). Lehoczky showed that the average schedulable utilization, for large randomly chosen tasks set, can increase from 88% to 95% range [14]. It is also important to note high utilization can be guaranteed by appropriate choice of task periods. RMA scheduling algorithm can schedule task up to 100% utilization when \( D_i = T_i \) and if the tasks period are harmonic that is each task period is an exact integer multiple of the next shorter period [14].

In designing software for a self-contained AMR system using behavior-based intelligent control, one of the frequently asked question is “what is the maximum number of behaviors that can be implemented in the robot and yet satisfies the timing requirements?”. The question can be answered by predicting an upper bound on the number of behaviors that can be implemented using the RMA utilization analysis. In order to facilitate this prediction, the following assumptions are made:

- The software architecture must follow behavior layer architecture that consists of a set of sensors, behaviors, actuators and a subsumption unit. An example of the architecture is illustrated in Figure 1.
- The component composition and implementation must follow the PECOS model. The AMR composition example is shown in Figure 2.

Generally, tasks in an AMR system can be divided into three groups according to real-time timing requirements: hard real-time requirement, firm real-time requirement and soft real-time requirement. The hard real-time task set represents the periodic processes that do not depend on specific AMR applications. Examples of such tasks are closed-loop motor control processes, and any process whose failure can leads to robot or human safety. In Figure 2 components motorctrl_right, motorctrl_left and MobileRobot are considered as hard real-time task. The firm real-time task set support the execution of the behavior process that determines the reactivity of the robot. In Figure 2 components Avoid, Cruise, Stop and Subsumption are considered as firm real-time tasks. The third task set is soft real-time, refers to processes that allow the representation of the middle and deliberative layers. In Figure 2 component manrobotintf is considered as soft real-time task.

From Theorem 1 sufficient schedulability tests of task under RMA defines utilization \( U_n \). Since the number of tasks and the computation time of the tasks depend on the number of sensors, actuators and behaviors in the behavior layer architecture, the following assumptions
are made to simplify the prediction of the upper bound on the number of behaviors:

- There are two groups of task periods: \( T_b \) is task’s period for firm real-time tasks and \( T_a \) is task’s period for hard real-time tasks.
- All sensors, actuators and actuator’s parent component belong to hard real-time task, while subsumption and all behavior components are belong to firm real-time task.

Therefore, the utilization of the AMR system is a function of number of robot behavior \( N_b \), number of actuator \( N_a \) and number of sensor \( N_s \):

\[
U(N_b, N_a, N_s) = \sum_{i=1}^{N_b} \frac{C_{bi}}{T_i} + \sum_{i=1}^{N_a} \frac{C_{ai}}{T_a} + \sum_{i=1}^{N_s} \frac{C_{si}}{T_s} + k
\]

(1)

where \( C_{bi} \) is the computation time of \( i \)th behavior component, \( C_{ai} \) is the computation time of \( i \)th actuator component, \( C_{si} \) is the computation time of \( i \)th sensor component, \( T_b \) is the task’s period for hard real-time task, \( T_a \) is the task’s period for firm real-time task and \( k \) is the sum of utilization for subsumption component and parent component which is a constant for a given implementation. From Eq. 1, it can be seen that, knowing the values of \( N_b, N_a \) and \( k \), an estimation of the upper bound for the number of robot behavior \( N_b \) for can be predicted.

### 2.4 Exact Feasibility Analysis

The utilization bound feasibility test given in Theorem 1 is widely recognized and frequently cited because it is simple in concept and computational complexity [15]. The feasibility test is referred as sufficient but not necessary or quite pessimistic. To enhance the previous prediction of the upper bound for the number of behaviors, a more exact test using response time schedulability analysis will be applied.

The goal of response time schedulability analysis is to show that each task invocation finishes before its deadline. For each invocation of a task \( i \), worst-case response time \( R_i \) is calculated and compare with deadline \( D_i \). Eq. 2 gives the total calculation for worst-case response time of a task \( i \) by adding its own worst-case computation time \( C_i \) plus the interference on how often higher-priority task \( j \) invocation can preempt an invocation of task \( i \) [16]. In the equation \( R_i \) appears on both sides of the equality. Fortunately, the equation can be solved iteratively where it’s begin with an initial approximation for \( R_i \) of 0, then the \( (x+1) \)th approximation can be defined in term of the \( x \)th one.

\[
R_i^{x+1} = C_i + \sum_{j \neq i \quad |p(j)|^x} \left( \frac{R_j^x}{T_j} \right) C_j
\]

(2)

The response time schedulability analysis allows the adjustment of the period for behaviors in an AMR system based on the result from Section 2.2, so that the critical behaviors can be ensured to not miss their deadlines, and the less critical behavior which does not effect the reactivity of the robot can be allowed to miss some of its deadlines. Some behaviors which belong to firm real-time tasks can be allowed to miss some of their deadlines in certain conditions.

The reactivity of AMR depends on the robot speed and the sensor task period. The relation between speed and task period is stated in the following equation [17]:

\[
T_i = \frac{k}{V_c} \quad \text{(3)}
\]

where \( T_i \) is the \( i \)th task period, \( k \), is a distance constant and \( V_c \) is the current robot speed. Eq. 3 can be used as a guide for selection of the speed and the task period.

### 3. Experimental Results

To evaluate the proposed approach in prediction of an upper-bound for the number of behaviors, an experiment involving an AMR real-time implementation was set up. The prediction made using the proposed approach is experimentally verified against the implemented AMR system. In this experiment the reliability and reactivity of the AMR behavior will be analyzed in the real AMR implementation given the temporal properties derived from the prediction. The AMR component-based software modeled in Figure 2 was implemented on a 16-bit AMD188ES microcontroller with memories of 64Kb ROM and 128Kb RAM. The AMR is shown in Figure 3.

Figure 3: The mobile robot.

The software tools used for the software development are Paradigm C compiler [18] for generating ROMable code and µC/OS-II real-time kernel [19] for multitasking support.

#### 3.1 Utilization Feasibility Analysis

Based on the measurements of computational times of the AMR system at runtime, it was found that: each behavior and subsumption components has a computation time of one millisecond, i.e. for all \( i \), \( C_{bi} = 1 \), and the computation time for the actuators (\( C_{ai} \) is 3 milliseconds each. Since, there are two actuators (\( N_a = 2 \) in this AMR, substituting the values of \( C_{bi} = 1 \), \( C_{ai} \) and \( N_a \) in Eq. 1, results in:

\[
U(N_b, 2, N_s) = \frac{N_b}{T_b} + \frac{6}{T_a} + \sum_{i=1}^{N_s} \frac{C_{si}}{T_i} + k
\]

(4)

Since, each behavior component is associated with a sensor component, \( N_i \) will be equal to \( N_b \). Thus, Eq. 4 simplify to:
The function $U(N_b,2)$ is analyzed for different reasonable values of $T_a$ and $T_b$ for the robot. These values were selected with respect to the closed-loop control requirements of the motors in the AMR system.

The setting of $T_a$ and $T_b$ to be analyzed in this experiment are listed in Table 1. Figure 4 shows that maximum number of allowable behaviors (where the utilization value is below 1) is determined by choice of $T_a$.

Table 1: Various setting for $T_a$ and $T_b$

<table>
<thead>
<tr>
<th>Setting</th>
<th>$T_a$</th>
<th>$T_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>S2</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>S3</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>S4</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>S5</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>S6</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>S7</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>S8</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>S9</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

In Figure 4, two lines of utilization bound are plotted to identify the sufficient condition for upper bound on the number of robot behavior. The first utilization bound is based on Lehoczky, (1990), is 100% utilization that is represented by darker solid black line when $u_1 = 1$ and the second utilization, based on Liu and Layland (1973), is $u_2 = n(2^T_2 - 1)$ is represented by darker dashed line. Utilization equal to 1 can be used in this experiment because $D_i = T_i$ and the tasks period are harmonic. However, task set, deadlines and tasks periods in future implementation of this AMR system may change, so general schedulability condition $u_2$ is much more applicable here.

Based on values of $T_a$ in Figure 4, the intersection of each value (three group: S1 and S2 for $T_a$ = 50, S3-S5 for $T_a$ = 20 and S6-S9 for $T_a$ = 10) with $u_2 = n(2^T_2 - 1)$ and

$$U(N_b,2) = \frac{N_b}{T_b^{\frac{1}{2}}} + \frac{6 + N_b}{T_a^{\frac{1}{2}}} + k$$ (5)

with $u_1 = 1$ are observed. Table 2 list the minimum intersection point with $u_2 = n(2^T_2 - 1)$ and maximum intersection point with $u_1 = 1$ for each group of $T_a$.

Table 2: Intersection point of $T_a$ group when $u_1$ and $u_2$

<table>
<thead>
<tr>
<th>$T_a$</th>
<th>$n(2^T_2 - 1)$</th>
<th>$u_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

The results from Table 2 can be used to estimate the maximum number of behavior that can be implemented in the AMR system. For example, with $T_a$ = 50, under RMA theorem the AMR used in this experiment can reliably control maximum 14 robot behaviors. Under condition when $D_i = T_i$ and tasks period are harmonic the behavior number can be increased up to 29.

Using the composition in Figure 2, a set of suggested temporal properties from Table 2 is implemented with three behaviors ($N_b$=3). The temporal properties for the MobileRobot components are tabulated in Table 3 with active component list and period for each component. The task component in the table are ordered from highest to lowest priority.

Table 3: MobileRobot’s temporal properties

<table>
<thead>
<tr>
<th>Component</th>
<th>Task Period (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileRobot</td>
<td>10</td>
</tr>
<tr>
<td>Motorctrl_left</td>
<td>10</td>
</tr>
<tr>
<td>Motorctrl_right</td>
<td>10</td>
</tr>
<tr>
<td>Subsumption</td>
<td>20</td>
</tr>
<tr>
<td>Stop</td>
<td>20</td>
</tr>
<tr>
<td>Avoid</td>
<td>20</td>
</tr>
<tr>
<td>Cruise</td>
<td>20</td>
</tr>
<tr>
<td>Manrobotintf</td>
<td>100</td>
</tr>
</tbody>
</table>

The experiment results show that the AMR can reliability and reactively react to its environment. From this result it can be concluded that the satisfaction of the RMA utilization bound feasibility test by AMR system enables the robot to perform stable behavior and met its functional requirements.
3.2 Exact Feasibility Analysis

To illustrate this exact analysis, the most critical task set from Table 2 with $T_i = 10$ and number of behavior equal to 3 were selected, i.e S9 setting in Figure 4. The set of behavior and task in Figure 2 can be used to illustrate this S9 setting. Response time analysis from RMA is then used to determine whether the deadline for each task in setting S9 can be met. Since, there are eight tasks derived from the AMR component composition, the analysis technique were applied for values of $i$ ranging from 1 to 8. For each $i$ worst-case response time $R_i$ was calculated and compare with deadline where $D_i = T_i$. The calculated $R_i$ for all task are listed in Table 4.

<table>
<thead>
<tr>
<th>Task</th>
<th>Period (ms)</th>
<th>WCET (ms)</th>
<th>$R_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileRobot</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>MotorCtrl_left</td>
<td>10</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>MotorCtrl_right</td>
<td>10</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>Subsumption</td>
<td>20</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Stop</td>
<td>20</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Avoid</td>
<td>20</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>Cruise</td>
<td>20</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Manrobotintf</td>
<td>100</td>
<td>16</td>
<td>-</td>
</tr>
</tbody>
</table>

Response time for task Manrobotintf is not listed because $R_i$ is greater then $D_i$, meaning that Manrobotintf task will miss some of its deadline in the execution of the task. From the observation on the AMR functionality of the LCD display, the performance of the display is stable and the display information reflecting the robot movement and status. This is because Manrobotintf task is belong to soft real-time requirement group which does not affect the robot behavior and reactivity of the robot system. The output produced from the missed deadline is still useful to be displayed.

Based on Table 4, adjustment on the period for behaviors in an AMR system can be made. Some behavior can be allowed to miss some of its deadline in certain condition. In this experiment the Cruise behavior is allowed to miss some of its deadline by changing period for Subsumption component to 15 milliseconds. However, under this condition the implementation result shows that the reactivity of the AMR still can be maintained.

From the experimental results it can be stated the adjustment of task’s period for firm real-time components can be performed based on response time analysis and the speed of an AMR.

4. Conclusion

The abilities to predict and analyze timing are key requirements for reliable mobile robot systems. A detail approach using RMA as an analytical model and PECOS model as component model to enable predictable of AMR real-time systems is proposed. The approach aims to address the issues of ERT software for resource constraint AMR and predictable real-time performance of the AMR system.

The proposed approach is illustrated using an experiment on a real-time implementation of AMR case study. From this analysis and experiments, it was illustrated how RMA scheduling is used to identify the number of behavior that can be implemented in the AMR system. Based on the number of behavior implemented, we show how response time schedulability analysis can be used to make more precise prediction and selection of the AMR behaviors’ period by analyzing the robot reliability and reactivity.

From the experiment results, it is concluded that high level prediction proposed here can avoid timing error in the field and costly late rework at implementation phases. This approach allows the robotic engineer to validate their design for timing correctness at high level using software engineering and real-time technologies.

5. Acknowledgements

The authors would like to thank Public Service Department of Malaysia for funding of this Ph.D. work.

References


