

Enhancements of PECOS Embedded Real-Time Component Model for Autonomous Mobile Robot Application

Dayang Norhayati Abang Jawawi, Safaai Deris and *Rosbi Mamat
Faculty of Computer Science and Information Systems
* Faculty of Electrical Engineering
Universiti Teknologi Malaysia, 81310 UTM, Johor, Malaysia.
dayang@fksm.utm.my

Abstract

Recently, Component-Based Software Engineering (CBSE) has becoming a popular approach for developing embedded software. In CBSE, a component model is required to specify the standards and conventions imposed on developers of components. Industrial component models such as CORBA, COM and JavaBeans are generally not suitable for embedded real-time (ERT) systems. Consequently, a number of component models suitable for CBSE of ERT software such as PBO, Koala, PECOS and ReFlex are introduced. Assessments of the PECOS component model were conducted to evaluate the suitability of PECOS component model for adoption in CBSE of autonomous mobile-robot (AMR) software. The assessments emphasize on three requirements: facilitates predictable real-time performance, support for resource constraint systems, and support platform-independent implementation. Three enhancements were proposed for the PECOS component model. These enhancements were implemented on a real two-wheeled mobile robot, and results show that, the PECOS component model together with the proposed modifications can generate application suitable for resource constrained AMR systems, the new mapping of component behavior to tasks process can be used to guarantee the run-time predictability and performance, and the new implementation framework proposed enable platform independent development of AMR software.

1. Introduction

Developing software for Autonomous Mobile Robot (AMR) is challenging since it requires knowledge in embedded systems, real-time software issues, control theories and artificial intelligence aspects. Component Based Software Engineering (CBSE) approach has been recognized as a way to alleviate these challenges. A component-based solution is expected to help robotic research groups in the following aspects [1]: 1) exchange of software parts or components between robotics laboratories, allowing specialists to focus on their particular field; 2) comparison of different solutions

would be possible from the available components; 3) startup in robot research can be accelerated using the available components; and 4) speed up the transfer of research laboratories works in robotics to commercial business application.

There have been some efforts on providing CBSE of robot software [2-4]; however, most of these works do not address the issues of embedded real-time (ERT) software requirements for resource constraint AMR and predictable real-time behavior of the robot. Furthermore, only platform-dependent components were considered in those works.

In CBSE, the choice of a component model is very important as it specifies the standards and conventions imposed on developers of components in order to achieve uniform composition, appropriate quality attributes and deployment of components and applications [5]. Industrial component models currently available such as OMG's CORBA Component Model (CCM), Microsoft's (D)COM/COM+, .NET, SUN Microsystems' JavaBeans and Enterprise JavaBeans, are generally complex, require large resources such as memory and computation power, and are platform dependent [6, 7]. 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 such as AMR.

Consequently, a number of component models such as ReFlex [8], PBO [9], Koala [10] and Pervasive Component Systems (PECOS) [11], 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. Evaluation of these component models against the industrial requirements of heavy vehicle sector shows that PECOS is the most complete component technology with good support for industrial requirements [12].

The main objective of this paper is to assess the PECOS component model for adoption in CBSE of AMR software with particular emphasize on three requirements: 1) facilitates predictable real-time performance of the components assembly; 2) support for resource constraint embedded AMR systems; and 3) support high-degree of platform-independence.

The layout of this paper is as follows. In Section 2, the PECOS component model is briefly described. In Section 3, the PECOS model is assessed using a case study of an AMR software development based on PECOS. As the results of the assessment, in Section 4, some enhancements to the PECOS component model are proposed and implemented on an actual AMR. Some results on the real-time implementation of are also described. Finally, the conclusion is presented in Section 5.

2. PECOS Component Model

PECOS component model was originally developed for field device systems. PECOS component consists of two main parts, i.e. the static structure model which describes the entities included in the model, their features and properties; and the execution model which defines the behavior of the component execution.

2.1. Static structure model


There are three main entities in the PECOS model: *components*, *ports* and *connector* [13]. Components are used to organize the computation and data into parts that have well-defined semantics and behavior. A port is a reference to data that can be read and written by a component and enables a component to be connected to another component through a connector. Data passed over the port is specified with name, type, range and direction (in, out or inout). Only compatible ports can be connected with connectors.

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 PECOS components can be hierarchically built from other subcomponents. A component which contains subcomponents or children is called a composite component or a parent component, and these subcomponents are not visible outside the composite component. A component without subcomponents is called a leaf component. A property bundle which describes the nonfunctional aspects such as timing or memory usage is associated with this static structure of component.

The components composition using PECOS static structural model is performed by connecting the compatible ports between components. Figure 1 illustrates integration of four components in PECOS for a motor speed control application. MotorSpeedControl is a composite component with three subcomponents, i.e. PI, Encoder, and Motor. Two connections in the composition are connection between output port *speed* to input port

actualSpeed and connection between output port *controlSignal* and input port *signal*.

Figure 1 also shows two active components: MotorSpeedControl and PI marked with “” at the upper right corner, and two passive components: Encoder and Motor.

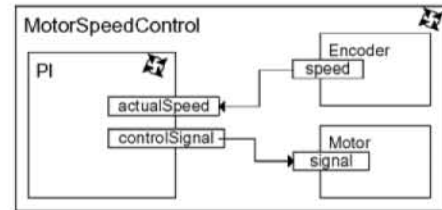


Figure 1. PECOS structure model for a motor speed control application

2.2. Execution model

There are two different behaviors associated with active and event components: *execution behavior* and *synchronization behavior*. Execution behavior determines the action that is performed when the component is executed while the synchronization behavior is responsible for synchronizing the data space of the active or event component with that of the parent [13].

In PECOS, data synchronization between two connected active or event components' ports is required to assure that data cannot get corrupted due to two simultaneous write operations from components in different thread. To solve this data synchronization problem, every active and event component is equipped with its own private data space where the data can be updated independently by the component. At specific intervals, this private data space is then synchronized its parent or composite component.

The execution behavior is based on the following runtime rules [13]: 1) The behavior of passive component is executed in the thread of its parent component; 2) Synchronization behavior for active and event components is executed in the thread of the parent component; 3) Active and event components execute their own execution behavior in their own thread of control; and 4) Every composite component has to provide a schedule for its children.

3. Assessment of PECOS Component Model

Software design using components involves connecting sets of component to create a software system capable of performing some useful function [14]. This activity consists of *component integration* and *component composition*. The goal the design activity is to find the most appropriate and feasible combination of the

component candidates. Based on the design, the *implementation* of the software is performed.

Assessment of PECOS component model is conducted during *component integration*, *component composition* and *implementation* stages. The design process of an AMR software case study using PECOS model is illustrated and the model is assessed on the following aspects: 1) facilitates predictable real-time performance of the components assembly; 2) support for resource constraint embedded AMR systems; and 3) support high-degree of platform-independence.

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 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 50 milliseconds using the proportional-integral (PI) control algorithm. The embedded controller also monitors the robot environment using four infrared (IR) proximity sensors and communicates with human using Liquid Crystal Display (LCD) and switches.

3.1. Component integration using PECOS

Components integration is the mechanical task of wiring components together by matching the needs and services of one component with the services and needs of others [14]. PECOS model supports component integration with static structure and execution model.

The components integration using PECOS static structural model is performed by connecting the compatible ports between components. Figure 2 illustrates integration of fifteen components in PECOS for the AMR system. The integration consists of seven active components and eight passive components. In Figure 2 two composite components are shown: *motorctrl_right* and *motorctrl_left*.

Figure 3 shows the hierarchical view of components and the private data spaces derived from integration in Figure 2. The data spaces used by each component during execution behavior (shown in solid arrow) and synchronization behaviors (shown in dotted arrow) are also illustrated in the figure.

Based on static structure and execution model, it can be concluded that component integration using PECOS model can explicitly describes the real-time behavior of a component using static structure and execution model. PECOS integration, using static structure and execution model also supports static binding of components at design-time composition; this allows optimization of the

design and code toward resource constraint of AMR systems.

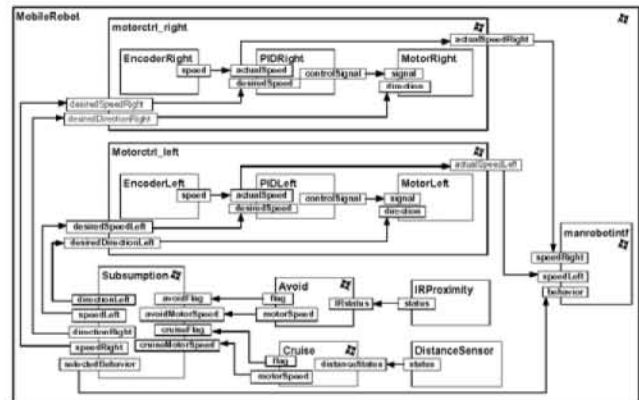


Figure 2. Components integration using PECOS model

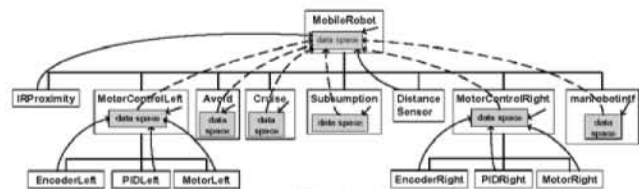


Figure 3. Hierarchical view of the AMR components

Currently, PECOS does not allow connection of constant value to input port in their static structure. We proposed to further enhance PECOS reusability by allowing connection of constant value to input port in their static structure.

3.2. Component Composition using PECOS

Once the real-time behavior and data synchronization has been specified in the integration stage, the next stage in PECOS assembly process is to consider the reasoning of the integration in component composition activity. Component composition supports means of determining the properties of assemblies in order to predict their runtime compatibility [14]. The properties of assemblies are specified using property bundle in PECOS model. The timing property in the property bundle can be predicted using Rate Monotonic Analysis (RMA) based on PECOS component composition approach [15].

RMA verification in PECOS is to check whether the entire components involved in the composition meet their deadlines. To use RMA to verify the component composition in Figure 2, a mapping of the composed components to RMA tasks is performed. In the original PECOS mapping of component behavior to tasks, the following mapping processes are performed: 1) passive component is treated as a periodic task and is associated

with period as defined by its parent; 2) active or event component is decomposed into two tasks: execution task and synchronization task, in which the period of the synchronization task inherits its parent period; and 3) mapping aperiodic behavior to periodic task using sporadic servers.

Based on the above PECOS mapping process, the resulting tasks for the MobileRobot components used in this case study are tabulated in Table 1. The tasks in the table are ordered from highest to lowest priority. The result from the table is used as input for RMA analysis of timing properties. The tasks worst-case execution times (WCET) and period are examples of property bundle in PECOS model. The tasks WCET are measured at runtime, and tasks periods are obtained from the AMR requirements.

Table 1. The result of PECOS component mapping for the AMR system

Task	Period (ms)	WCET (ms)
EncoderRightexec	50	0.64
PIDRightexec	50	0.48
motorctrl_rightexec	50	19.20
EncoderLeftexec	50	0.64
PIDLeftexec	50	0.48
motorctrl_leftexec	50	19.20
IRProximityexec	50	0.03
DistanceSensorexec	50	0.70
Avoidsync	50	0.02
Cruisesync	50	0.02
Subsumptionsync	50	0.02
motorctrl_rightsync	50	0.02
motorctrl_leftsync	50	0.02
manrobotintfsync	50	0.02
Avoidexec	300	17.00
Cruiseexec	300	0.02
Subsumptionexec	300	0.04
manrobotintfexec	500	16.00

Theorem 1 from RMA [16] is then used to determine whether the deadline for each task in Table 1 can be met. Since, there are eighteen tasks derived from the AMR component composition, the theorem was applied for values of i ranging from 1 to 18. For each i at least one possible pair of (k, l) that satisfies the equation in the theorem was found. This indicates that the design composition of components in Figure 2 is predicted to be schedulable according to RMA theory. Thus, PECOS approach and real-time theory such as RMA can be used to guarantee the AMR system predictability and performance.

Theorem 1: A set of n independent periodic tasks will always meet its deadlines, for all task phasing, if and only if

$$\forall, 1 \leq i \leq n, \min_{(k,l) \in R_i} \sum_{j=1}^i C_j \frac{1}{IT_k} \left\lceil \frac{IT_k}{T_j} \right\rceil \leq 1$$

where C_i , and T_i , are the WCET and period of task i , respectively, and

$$R_i = \{(k, l) \mid 1 \leq k \leq i, l = 1, \dots, \left\lfloor \frac{T_l}{T_k} \right\rfloor\}.$$

PECOS also uses constraint solving approach to perform the schedule verification, since, it is claimed that the timing verification using RMA alone is not enough when not all tasks run concurrently and hence schedule verification is needed to check the possibility to fit execution and synchronization behavior sequentially in each task [15]. Our experiment with PECOS shows that the use of RMA alone is enough to reason about the component composition due to the way the event components are mapped to active components in a time triggered execution such that all tasks run concurrently. This is further discussed in Section 4.2.

3.3. Implementation of PECOS model

To support the implementation of PECOS component model, the PECOS consortium has developed some tools such as Component Composition (CoCo) description language for specifying components, code generator for generating Java/C++ code skeletons from CoCo and runtime environment (RTE) which interfaces generated code from the real-time operating system used. However, many of these tools are incomplete and information on the RTE is not publicly accessible as it is a proprietary of the ABB Company [17].

After experimenting with the PECOS component model in the AMR application and implementing it, it is found that the assembling activities of components at design stage are simple but the implementation of the components is complex without the support from PECOS tools. With the absence of PECOS supporting tools and RTE, there are at least three options in which the PECOS component model can be used effectively in the AMR software development: 1) develop our own supporting tools and RTE; 2) seek alternative from other existing tools such as real-time Unified Modeling Language (UML) tools [18]; or 3) propose an alternative implementation framework for CBSE of AMR embedded software development.

Option three has been selected in this work, where a simple implementation framework is proposed and has been used to implement PECOS component model for component based development of AMR embedded software.

4. Enhancements of PECOS Component Model and an Implementation Framework

Based on the assessment conducted in Section 3, three modifications or enhancements were identified for the PECOS component model for adoption in AMR embedded software development. These proposed modifications or enhancements are in the aspects of: 1) allowing connection of constant values to input ports; 2) mapping component behavior to tasks and 3) new simple implementation framework.

4.1. Allowing constant connection to input ports

The components integration using PECOS structural model is performed by connecting the compatible input and output ports between components. To increase reusability and flexibility in PECOS model, connection between input ports with constant values is allowed. This gives flexibility for testing and debugging of components and also useful for AMR software to reconfigure components for use with specific hardware or application.

As an example, consider motor control components such as *motorctrl_right* from the AMR application case study as shown in Figure 4. Depending on the type of

motor and the algorithm for calculating the motor control signal, the connection of PIDSetting input port (IP02) for motor control component will vary. By allowing constant values connection, the PIDSetting input port can be connected to other output port or constant values. Figure 4 shows an example of connecting PIDSetting input port (IP02) from *motorctrl_right* with constant values of 1000, 20, 0 and 50.

4.2. Mapping of component behavior to tasks

In our integration process only passive and active components are supported, and each event component is converted to an active component with cycle time determined experimentally.

This mapping was applied to the MobileRobot components used in this case study. The results are tabulated in Table 2, where the tasks in the table are ordered from highest to lowest priority. The results from Table 2 are used as input for RMA analysis of timing properties using Theorem 1. It can be shown that, this new mapping process also resulted in a composition of components which is predicted to be schedulable according to RMA theory.

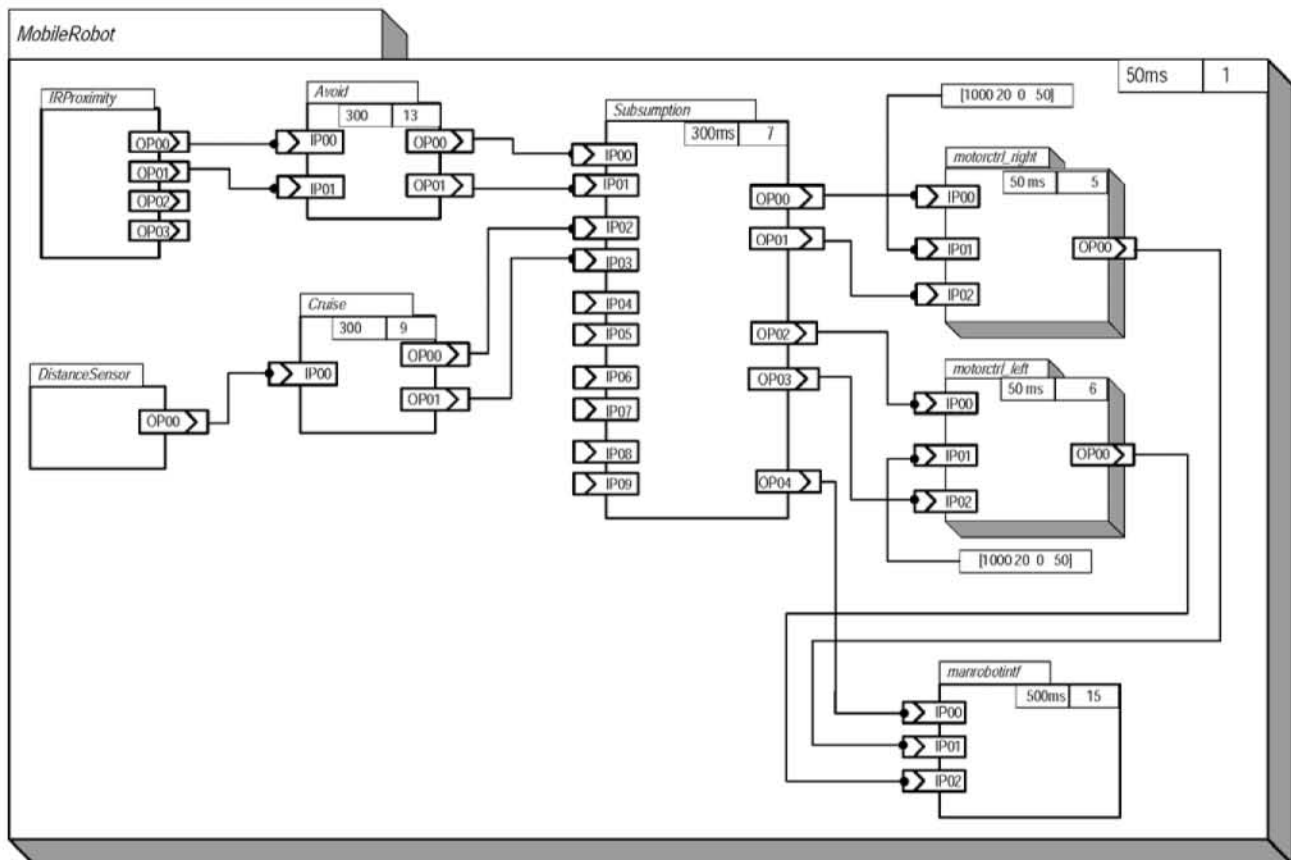


Figure 4. Components integration of an AMR application case study

The MobileRobot component of Figure 4 is mapped to a task which executes sequentially two passive components IRSensor and DistanceSensor; and six synchronization parts of its child components, i.e. Avoid, Cruise, Subsumption, motorctrl_right, motorctrl_left and manrobotintf. Note that, in the original PECOS mapping these six synchronization parts are mapped to six tasks, and the two passive components are mapped to two tasks. Thus more tasks results from the original PECOS mapping. As shown in Table 2, a smaller number of tasks (seven tasks) results from this new mapping process compare to the mapping process originally proposed by PECOS (eighteen tasks) as tabulated in Table 1. This has an advantage in our AMR application, since, small number of tasks means less memory is required for context storage during preemption, making this mapping process suitable for memory constrained embedded systems. However, this has a small disadvantage, since, the synchronization of all child tasks are handled by parent component, the whole software become less reactive. In our implementation this is solved by assigning the parent component with higher priority than its child tasks.

Table 2. The result of proposed component mapping for the AMR application

Task	Period (ms)	WCET (ms)
MobileRobot	50	0.87
motorctrl_right	50	20.40
motorctrl_left	50	20.40
Avoidexec	300	17.00
Cruiseexec	300	0.02
Subsumptionexec	300	0.04
manrobotintfexec	500	16.00

4.3. A new implementation framework

This new framework was originally intended for development of ERT software for AMR systems. The target audience is the mechatronic and robotics researchers which are not from software engineering background and do not have extensive programming experience. Experience shows that, however, the new framework is generally suitable for developing reactive embedded systems which typically use 8-bit or 16-bit microcontrollers with memory constraints and developed with C language.

In this framework, instead of using the CoCo language, graphical visualization of components is use for components definition and composition. Currently, codes have to be written manually from this graphical visualization. The graphical composition environment and

code generation tools based on the graphical visualization are part of our on going work. Figure 4 shows block diagram integration of the AMR case study using the proposed graphical visualization. The input ports and output ports are defined in the block. Each port is numbered accordingly for easy reference. Arrows on the ports indicate the direction or whether they are inports or outports. Bidirectional ports are not allowed in this framework. In the figure, composite components are shown by blocks with shadow and components with period and priority fields are active components and components without the filed are passive components.

The framework target for C language since, optimized C compilers are available for most micro-controllers, and C is portable across many platforms. Furthermore, C is a familiar language and has been use extensively by the target audience. The dependency on RTE is minimized by targeting the codes to standard calls for minimal real-time kernel. Consequently, this will enable platform independent development of AMR software.

The results of component integration and component composition are the creation of a configuration file **pecos_cfg.h** for inclusion in the main program. This file specifies the components to be included in the project and periods of execution and priorities of active components. The effect of this file is to include only the required components source codes in the final code and to configure periods and priorities of active components for the real-time kernel use. The code as follows shows a fragment of the configuration definitions in the **pecos_cfg.h** file:

```
// ***** CONFIG PECOS COMPONENTS :
// REQUIRE (1) or NOTREQUIRE (0)

#define COMP_MOTOR_REQ 1
#define COMP_IRPROX_REQ 1
#define COMP_DISTSENSOR_REQ 1
#define COMP_SUBSUMPTION_REQ 1
#define COMP_BEHAVIOR_BASED_CTRL_REQ 1
#define COMP_MANROBOT_INTRF_REQ 1
#define COMP_MOTOR_CTRL_REQ 1

// ***** CONFIG REAL-TIME PARAMETERS FOR ACTIVE
//COMPONENTS *****
//-- MANROBOTINTF
#define MRI_PERIOD 500
#define MRI_PRIO 15

//-- MOTORCTRL_RIGHT
#define MOTORCTRL_RIGHT_PERIOD 50
#define MOTORCTRL_RIGHT_PRIO 5
:
```

The connections between the compatible input ports and output ports are achieved by assignments of data defined by the components interface. For example the following code shows how connection between the output ports of Avoid component and the input ports of IRProximity component are made:

```

/-- Connect all inports to outputs
Avoid.IP00 = IRProximity.OP00;
Avoid.IP01 = IRProximity.OP01;

```

As shown in Figure 4, the top level component in this composition is the `MobileRobot` component which has the period of execution of 50 ms and priority of 1, i.e. the highest priority. Thus, the main program for this composition is the execution behavior of `MobileRobot` component. The template for this main file is shown in Figure 5.

```

#include "pecos_cfg.h"
#include "pecos.h"
// TOP LEVEL COMPONENT
void MobileRobot (void* data) {
/***** INITIALIZATION PART *****/
  /-- initialize all child components
  /-- all ACTIVE tasks will be created
  INITmanrobotintf();
  INITmotorctrl_right();
  INITmotorctrl_left();
  :etc
  for (;)
/***** EXECUTION PART *****/
  /-- Execute all passive components
  /-- Active components already executed
  EXECIRProximity();
  :etc
/***** SYNCHRONIZATION PART *****/
  /-- Synchronous all inports & outputs
  SYNCmanrobotintf();
  :etc
  /-- Connect all inports to outputs
  Avoid.IP00 = IRProximity.OP00;
  Avoid.IP01 = IRProximity.OP01;
  Subsumption.IP00 = Avoid.OP00;
  Subsumption.IP01[0] = Avoid.OP01[0];
  :etc
  /-- call RTK to create periodic execution
  OSTimeDelay (MOBILEROBOT_PERIOD);
}
void main(void) {
  /-- initialize hardware dependent parts
  /-- initialize Real-time kernel
  /-- create MobileRobot task
  /-- start the Real-time kernel
  while (1) ; // run endlessly
}

```

Figure 5. The code template for MobileRobot component task execution

4.4. Implementation Results

Following the modifications previously described and PECOS component definitions, the AMR software composed as in Figure 4 was implemented on a real two-wheeled-mobile robot with behavior-based intelligence system. The target board consists of a 16-bit AMD188ES microcontroller with 64Kb ROM and 128Kb RAM. The software tools used for the software development are

Paradigm C compiler [19] for generating ROMable code and μ C/OS-II real-time kernel [20] for multitasking support. The total code size for the resulting application is about 21Kb with RAM usage of about 15Kb. This indicates that, the PECOS component model together with the proposed modifications can generate application with minimal memories requirements, suitable for other resource constrained embedded systems.

The implementation also shows that the new mapping of component behavior to tasks process can be used to guarantee the AMR system predictability and performance. The process is simpler and the number of tasks obtained is reduced because the synchronization and passive parts of the component are executed sequentially in their parent execution task. Therefore, the used of constraint solving approach as proposed by PECOS consortium to check the possibility to fit execution and synchronization behavior sequentially in each task is not required in our component composition activity.

5. Conclusion

Assessments of the PECOS embedded real-time component model were conducted to evaluate the suitability of PECOS component model for adoption in CBSE of AMR software. The assessments were conducted on an AMR case study with emphasize on three important requirements: 1) facilitates predictable real-time performance of the components assembly; 2) support for resource constraint embedded AMR systems; and 3) support high-degree of platform-independence.

Based on the assessments conducted, three modifications or enhancements were identified for the PECOS component model. The enhancements proposed are in the aspects of: 1) allowing connection of constant values to input ports; 2) mapping component behavior to tasks and 3) new simple implementation framework.

These enhancements were implemented on a real two-wheeled mobile robot with behavior-based intelligence system. Results show that, the PECOS component model together with the proposed modifications can generate application with minimal memories requirements, suitable for resource constrained embedded systems. The implementation also shows that the new mapping of component behavior to tasks process can be used to guarantee the AMR system predictability and performance, and the new implementation framework proposed enable platform independent development of AMR software by removing the dependency on run-time environment, targeting the codes to standard calls for minimal real-time kernel, and the use of C language.

10. References

- [1] A. Oreback. "Components in Intelligent Robotics", *MRTC report ISSN 1404-3041 ISRN MDH-MRTC-15/2000-1-SE Component Based Software Engineering - State of the Art*, Mälardalen Real-Time Research Centre, Mälardalen University, January 2000, pp. 233-243.
- [2] S. Blum. "Towards a Component-based System Architecture for Autonomous Mobile Robots", *Proc. IASTED International Conference Robotics and Applications (RA'01)*, Tampa, 2001, pp. 220-225.
- [3] E. Messina, J. Horst, T. Kramer, H. Huang, and J. Michaloski. "Component Specifications for Robotics Integration", *Autonomous Robots*, 6, 1999, pp. 247-264.
- [4] iRobot Corporation, *Mobility Robot Integration Software User's Guide*, 2002.
- [5] F. Bachman, L. Bass, S. Buhman, L. S. Comella-Dorda, R.C. Seacord, and K. C. Wallnau. "Technical Concept of Component-Based Software Engineering". *Technical Report CMU/SEI-2000-TR-008*, Software Engineering Institute, Carnegie Mellon University, 2000.
- [6] U. Rasthofer, and F. Bellosa, "Component-based Software Engineering for Distributed Embedded Real-time Systems", *IEE Proceeding- Software*, 148(3). 2001, pp. 99-103.
- [7] F. Lüders, "Adopting a Software Component Model in Real-Time Systems Development", *Proceedings. 28th Annual NASA Goddard Software Engineering Workshop*, 2003, pp. 114 – 119.
- [8] A. Wall. "Architectural Modeling and Analysis of Complex Real-Time Systems", *Ph.D. Thesis of Mälardalen University*, 2003.
- [9] D. B. Stewart, R. A. Volpe, and P. K. Khosla. "Design of Dynamically Reconfigurable Real-time Software Using Port-Based Objects", *IEEE Transaction on Software Engineering*, 23(12), 1997, pp. 759 –776.
- [10] R. Ommering, F. Linden, J. Kramer, and J. Magee, "The Koala component model for consumer electronics software", *IEEE Computer*, 33(3), 2000, pp. 78 –85.
- [11] O. Nierstrasz, G. Arévalo, S. Ducasse, R. Wuyts, A. Black, P. Müller, C. Zeidler, T. Genssler, R. van den Born. "A Component Model for Field Devices", *Proceedings First International IFIP/ACM Working Conference on Component Deployment*, Springer-Verlag Heidelberg, Berlin, 2002, pp. 200-209.
- [12] A. Möller, Åkerholm M., Fredriksson J., Nolin M., "Evaluation of Component Technologies with Respect to Industrial Requirements", *Proceedings of the 30th EUROMICRO Conference on Component-Based Software Engineering Track*, Rennes, France, August 2004, pp. 56-63.
- [13] Genssler T. et. al., "PECOS in a Nutshell", *Technical Report*, PECOS Project, September 2002.
- [14] Crnkovic, I., and M. Larsson, *Building Reliable Component-based Software Systems*, Artech House, 2002.
- [15] R. Wuyts, S. Ducasse and O. Nierstrasz. "A Data-centric Approach to Composing Embedded, Real-time Software Components", *The Journal of Systems and Software*, 74, 2005, pp. 25-34.
- [16] Klien, M., T. Ralya, B. Pollak and R. Obenza, *A Practitioner's Handbook for Real-time Analysis*, Kluwer Academic Publisher. 1993.
- [17] B. Bouyssounouse, J. Sifakis, "Embedded Systems Design: The ARTIST Roadmap for Research and Development", *Lecture Notes in Computer Science*, Volume 3436 / 2005, Springer-Verlag GmbH, 2005, pp. 160-194
- [18] Douglass, B. P., "Real-Time UML". Addison Wesley. 2004.
- [19] Paradigm Systems, *Paradigm C++ Reference Manual Version 5.0*, Endwell, 2000.
- [20] Labrosse, J. J., *MicroC/OS-II The Real-Time Kernel*, 2nd edition, R&D Books, USA , 1999.