Proceedings

International Symposium on Control and Automation

CA 2008

13-15 December 2008

Sanya China

CPS Conference Publishing Services

Los Alamitos, California

Washington • Tokyo

IEEE computer society

Synchronization of Delayed Chaotic Neural Networks by LMI 55
Ranchao Wu and Weiwei Zhang
Multi-time Step Traffic Queue Prediction Model Based on Discrete Time Point Process (DTPP) 60
Wonho Kim and Byung-Ho Park
Control Architecture Design for an Gas Cutting Robot 66
Kisung Yoo
Automatic Coil-Handling Crane Control System 72
Chintae Choi
Development of Automation Strapping Machine Using PET Band 76
Kisung Yoo and Chintae Choi
Fault Diagnosis System for Turbo-Generator Set Based on Self-Organized Fuzzy Neural Network 78
Ping Yang and Zhen Zhang
Evaluating Employee Responses to the Lean Enterprise System at a Manufacturing Company in Cape Town, South Africa 85
Bingwen Yan and Keith Jacobs
Semi-automatic Software Development Based on MDD for Heterogeneous Multi-joint Robots 93
Hyun Seung Son, Woo Yeol Kim, and Robert Young Chul Kim
TDoA Based UGV Localization Using Adaptive Kalman Filter Algorithm 99
Wookjin Sung, Sungok Choi, and Kwanho You

Author Index 104

2008 Second International Conference on Future Generation Communication and Networking Symposia

Semi-Automatic Software Development Based on MDD for Heterogeneous Multi-Joint Robots

Hyun Seung Son[1], Woo Yeol Kim[1], R. Young Chul Kim[1,2]
[1]Dept. of Computer & Information Comm., Hongik University, Korea
{son, john, bob}@selab.hongik.ac.kr[1]
[2]Dept. of ITM, Illinois Institute of Technology, Wheaton, IL USA

Abstract- **It might be necessary to develop the multi-joint robots that can work very dedicated movable like a kind of unmanned ground vehicles on very complicated and hazardous environments. These typed robots might be difficultly so controlled and developed that they should control plenty of servo-motors and also diverse hardware. It will be very hard to reuse the software source codes associated with these own robots, much less for heterogeneous ones. It is very difficult to develop heterogeneous multi-joint robots. So we propose the semi-automatic software development based on MDD (Model Driven Development) for heterogeneous multi-joint robots.**

Keyword- Model Driven Development, CASE Tool, Multi-Joint Robot, Unified Modeling Language, Modeling

I. INTRODUCTION

In many fields, it is in progress to research the diverse typed robots such as robot appliances and toys, domestic droids, robot cars, military ones, micro robots.

Daniel Wilson [1] mentions that most approach of the general robots is the sense-think-act mechanism - the closed-loop process such that a robot senses the environment, thinks about what to do, and then acts in physical world. It is very similar to how we humans interact with world. The loop requires sensors to gather information, artificial intelligence to decide on a course of action and end effectors to interact with environment.

It uses actuators to move their arms, legs, or very sensitive tentacles. But these actuators are electromechanical motors to convert electricity to physical force and step motors to be used for precise movements.

To consider the movements of robot, we broadly classify the legged / the wheeled mobile movements.

On the wheeled styled movements of Robot, it is easily controlled, but just moves on a flat ground region. One the legged styled movements, it is possible to move any places with simultaneously controlling plenty of motors. The disaster rescue or military robot should even move in any dangerous environments. We will focus on the legged style Robot due to this bad situation. This paper is mentioned to the modeling tool for quickly and efficiently developing the software of multi-joint robot that is very difficult to develop with the traditional method.

In software engineering fields, MDA(Model Driven Architecture)[2,7,9] focuses on a mechanism that does the code generation via platform-dependent model through platform-Independent model for multi-platforms. We adopt one kind of software process mechanisms, which is MDA, into embedded software development, but should have made the automatic mechanism to convert on each step. Our first attempt defines the embedded software process. Based on this process, we make the automatic mechanism for heterogeneous multi-joint robots that easily develop through automatically converting from a model to the other model based on each particular hardware target. With this method we may enhance the code productivity associated with reuse of models.

Therefore, it may be possible to enhance the code development as we apply multi-joint robot development with MDD. It will also reduce the development cycle with reusing design/model. Due to reusable mechanism we can also develop software which is the hardware- independent [3,13].

This paper is suggested the development mechanism of multi-joint robot and introduced to the automatic development tool based on the mechanism.

With this tool we show to generate each source code based on heterogeneous multi-joint roots.

This paper is organized as follows: In section 2, we describe show our embedded MDD approach for developing embedded s/w system. In section 3, it shows the modeling example of heterogeneous multi-joint robots used in this paper. In section 4, it mentions the semi-automatic software development with the example of heterogeneous multi-joint robots. Finally we make a conclusion.

II. RELATED WORKS

A. A Modeling Approach based on UML

Our modeling approach focuses on the embedded software development. So, we don't need to use a lot of diagrams like UMLx.x. We follow the basic UML mechanism of Rational Rose, but not use all their diagrams for modeling and generating Code. We choose several diagrams for modeling embedded software, change and add new concepts into our chosen diagrams. Our diagrams like UMLx.x consist of static modeling and dynamic modeling. The static model uses class diagram to represent the static aspect of a system. In dynamic model, concurrent message

978-0-7695-3546-3/08 $25.00 © 2008 IEEE
DOI 10.1109/FGCNS.2008.132

diagram and concurrent state diagram represent the dynamic concurrent behavior of the system [11].

B. Model Driven Development (MDD) Approach

We develop the automatic MDD Transformation Process [10,12] based on our embedded MDA mechanism (that is, on hardware target) into which the original MDA concept (that is, on platform) are adopted. This process consists of TIM (Target independent model) stage, TSM (Target specific model) stage, and TDC (Target dependent code) stage. TIM defines a general model independent of the particular target domains. After requirement analysis, we may design TIM with extended xUML[4,5,11] and UML profile[3] at this TIM stage. TSM defines a specific model dependent of the OS, and hardware within the particular target domains [12]. At this TSM stage, we can model more complete with additional functions on TSM automatically transformed with TIM. TDC defines the source code per the specific target system. At TDC stage we can automatically generate codes whatever we need such as Java, C++, or C. Finally we can port complete executable code(s) into the target system(s). Now, we can semi-automatically develop the heterogeneous embedded software on the whole automatic MDD transformation process.

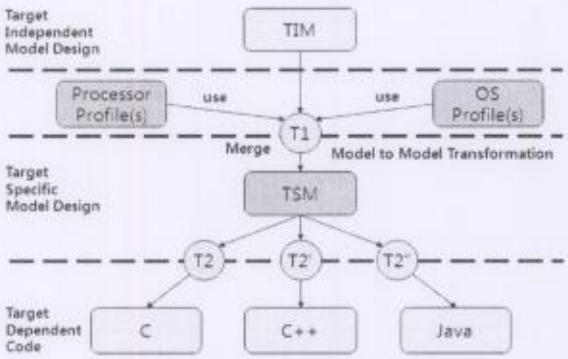


Figure 1. The automatic MDD Process [12]

Our MDD Transformation Process consists of two transformation steps T1, T2(or T2', T2'') in figure 1.

The first transformation T1 is automatically transformed TIM (which merges with the predefined Processor profile and OS profile) into TSM. The second transformation T2 is automatically generated the right executable TDC per each different target system. Its possible generable code languages are C, C++, and Java. The code generating method is automatically generated using conversing text from model [6,8,12].

Transformation T1: Transforming from a TIM to TSM(s)

associate with operating system and processor. The user will possible develop a system having only knowledge of services without detail implementation. Operating System consists of APIs of OS.

Transformation T2: Transforming from TSM(s) to a TDC(s)

transform the actually executable code from TSM through T1 transformations, we should do execute T2 transformations. T2 is generated a language with meta-template model on the basic of the class diagram, concurrent message diagram, and concurrent state diagram of TSM.

III. AUTOMATIC MDD MODELING MECHANISM

A. UML Profile

This UML profile is a important part of the modeling mechanism for generating source code.

The UML profile part consists of information about Middleware, RTOS, and Processor(s) such as the name and supportable RTOS of middleware, the name and GPIO image of processors, and a list of API functions be transformed.

Figure 2 shows the structure of meta profile with XML. We can get this information from the storied UML Profile to automatically generating the right code.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:complexType name="targetTagType">
      <xs:sequence>
         <xs:choice>
            <xs:element name="startTag" type="xs:string"/>
            <xs:element name="endTag" type="xs:string"/>
         </xs:choice>
         <xs:element name="contents" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="processorType">
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="img" type="xs:anyURI"/>
         <xs:element name="information" type="xs:string"/>
         <xs:element name="targetTag" type="targetTagType" minOccurs
="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="rtosType">
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="processor" type="processorType"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="MiddlewareType">
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="rtos" type="rtosType" maxOccurs="
unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="MetaProfileType">
      <xs:sequence>
         <xs:element name="Middleware" type="MiddlewareType"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="MetaProfile" type="MetaProfileType"/>
</xs:schema>
```

Figure 2. The structure of Meta Profile

Figure 3 shows the selection box (such as Middleware, RTOS, and Processor) of UML profile in the automatic embedded software development tool. Within the tool it displays information of selecting what target builds This means to automatically convert TIM(target independent model) into TDM (target dependent model), that is, what target builds.

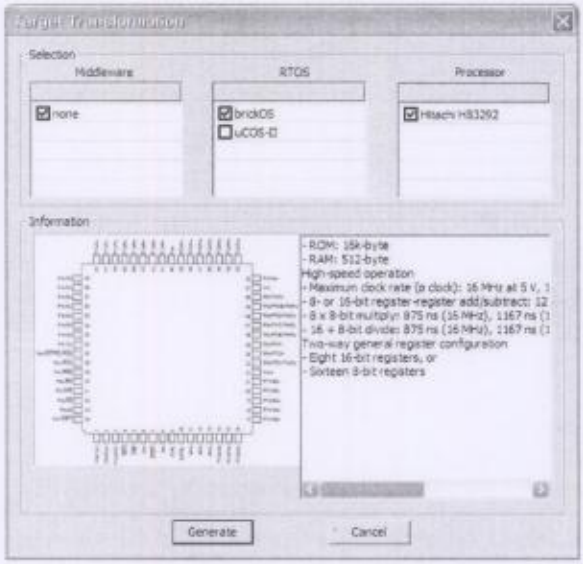


Figure 3. The selection box of UML profile

B. *Codable Templates from UML Diagrams*

Each diagram in the automatic tool consists of each independent file. Automatically to generate its related code based on it, we should map and load the data information of each diagram into "meta template model". For producing embedded software code, we use just three diagrams such as class diagram, concurrent message sequence diagram, and concurrent state diagram. It is integrated with "meta template model" with the data information of these three diagrams. In final step, to produce code should be converted these data into text file.

Class Diagram

We map the class diagram into meta template model. Figure 4 shows to associate each part of class diagram with each related part of meta template model with the dotted arrows. The name of class in diagram is stored by the unique "Class Name". The inheritance relationships between classes are loaded into "Parent List". The interface relationships between classes are loaded into "Interface List". The association relationships are also loaded into "the Association List". Attributes and functions are also loaded into "Attribute List" and "Function List" on Meta template model.

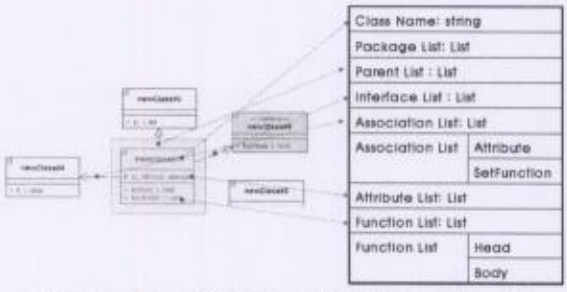


Figure 4. Mapping Class diagram into Meta template model

Concurrent Message Sequence diagram

This diagram is played an important role in determining how many objects will be created or called. Figure 5 shows to fill the implementation part of methods from the association relationship among objects. But it will be possible to change the generated code based on this diagram because it will be determined the final these methods through the concurrent state diagram. Therefore, it will be completely integrated the generated code from this diagram with the concurrent sate diagram.

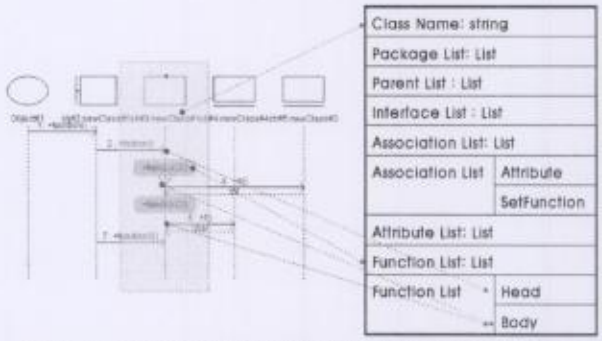


Figure 5. Mapping Concurrent Message Sequence diagram into Meta template model

Concurrent State Diagram

We just use to make the structure of the whole program code with other diagrams. But this diagram is very important part for generating the code for an embedded software development. This diagram(figure 6) can be directly mapping actual code and filling the detail information into meta template model.

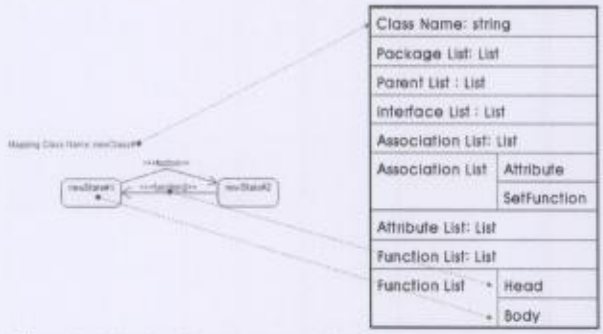


Figure 6. Mapping Concurrent state diagram into Meta template model

The Generated Code Template Based on Meta Template Model

The generated code template is automatically converted based on Meta template models are generated. Figure 7 shows to link the right places in the code template with the Meta template diagram.

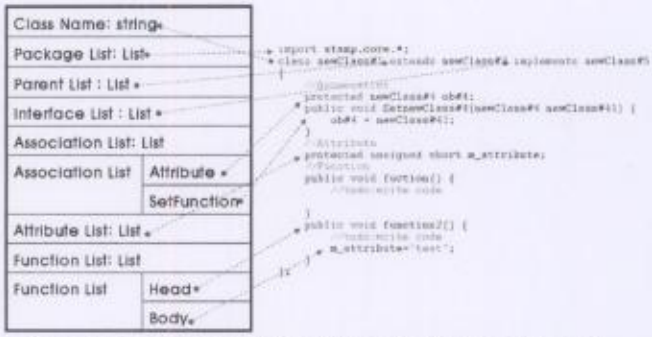
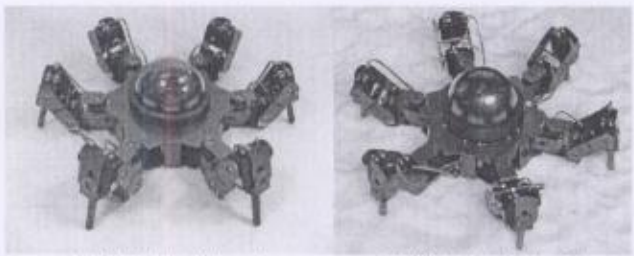


Figure 7. The generated Code Template based on Meta template model

IV. SEMI-AUTOMATIC SOFTWARE DEVELOPMENT

In figure 8, we use two heterogeneous multi-joint robots with 6 arms as one example in this paper.



(a) Multi-Joint Robots 1 (b) Multi-Joint Robots 2

Figure 8. The heterogeneous multi-joint robots

We installed different processor into each robot to test our approach. Table I mentions hardware information of two heterogeneous robots. (a) Robot installs 8 bits-Atmega128 and programs C language, and (b) Robot uses 8 bits-Ubicom SX48AC and embeds Java Chip for executing Java. Both of them use 18 motors to control multi-joints through communicating with Bluetooth.

TABLE I. THE HW CONF. INFORMATION OF HETEROGENEOUS ROBOTS

	(a) Multi-Joint Robots 1	(b) Multi-Joint Robots 2
Microcontroller	Atmel Atmega128 16MHz	Ubicom SX48AC 20MHz
OS	none	Javeline
RAM	4KByte	32 KByte
EEPROM	128KByte	32 KByte
Sensor	2	2
Communication	Bluetooth	Bluetooth
Motors	18	18
JVM	No	On Hardware
Languages	C	Java

A. TIM(Target independent model) Phase

It is just software design for modeling target independent model. In this step, there are two diagrams such as the static model and the dynamic models.

1) The Static Modeling

This model uses to design the static structure of an embedded system. Figure 9 shows the class diagram of multi-joint robot. The class diagram includes sensors, controller, and data storage. The sensors may consist of ultrasonic, vibration, temperature, and magneton sensors. In this time, we just install ultrasonic sensor, but can attach more sensors.

Each arm of multi-joint robot has three servo motors, and consists of classes to automatically set whatever position each arm changes. Also the data is stored in "DataRepository" Class.

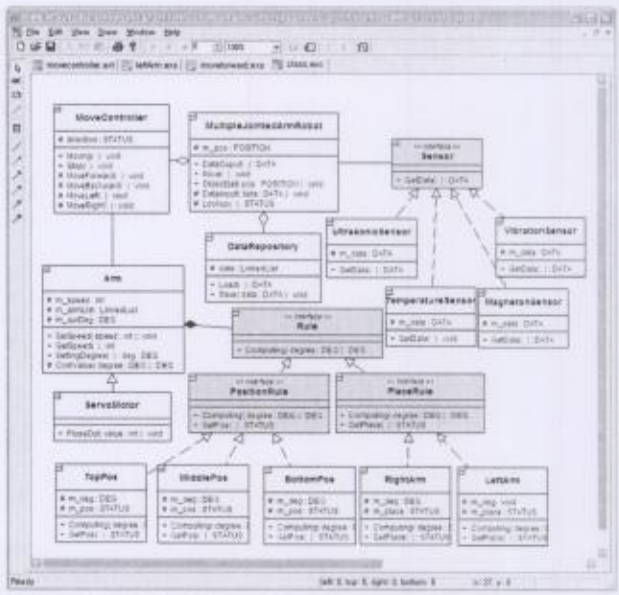


Figure 9. The class diagram of a multi-joint robot on target independent model

2) The dynamic modeling

This modeling includes Concurrent Message Diagram and Concurrent State Diagram. Concurrent Message Sequence Diagram is extended to assign role, and to add synchronism.

This also is modeling for representing interaction between objects. Figure 10 shows the Concurrent Message Diagram of a multi-joint robot on target independent model. For example, the sensor object gets data from an event outside, and sends the data to the controller object. After analyzing the data, the controller object that uses "SettingDegree()" method sends a message to 6 arms simultaneously.

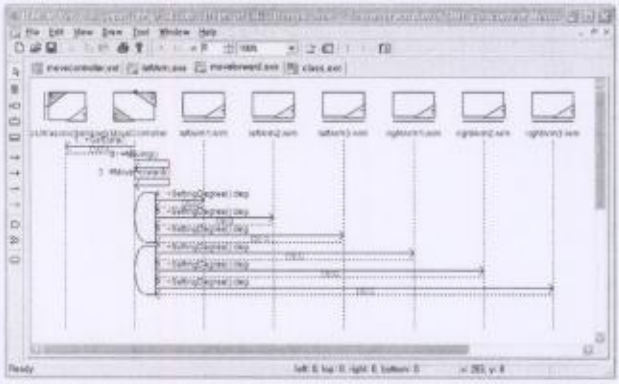


Figure 10. The concurrent message sequence diagram of a multi-joint robot on target independent model

Concurrent state diagram represents to change the behavior of the control object out of embedded system, that is, robot. Figure 11 shows the state of multi-joint robot in the concurrent state diagram such as the moving state and the stop state. The moving state can change the front, the rear, the left, and the right movement from the input value of the sensor.

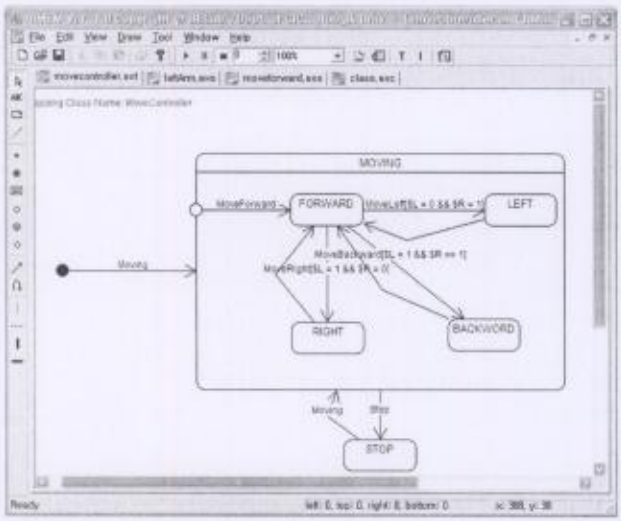


Figure 11. The concurrent state diagram of a multi-joint robot on target independent model

B. *TSM(Target specific model) Phase*

TSM phase should convert the target independent model into the target dependent model, which is applied with profiles of middleware, RTOS, and processors. It converts TIM into TSM automatically through our developed tool. In figure 12, 13, it shows to convert a TIM into each different TSM according to professor, RTOS, and middleware. Figure 12 shows to click each 'none' button on Middleware and RTOS, and chooses Atemega128 on processor for Multi-Joint Robots 1. Then clicks 'Generate' button to automatically generate TSM.

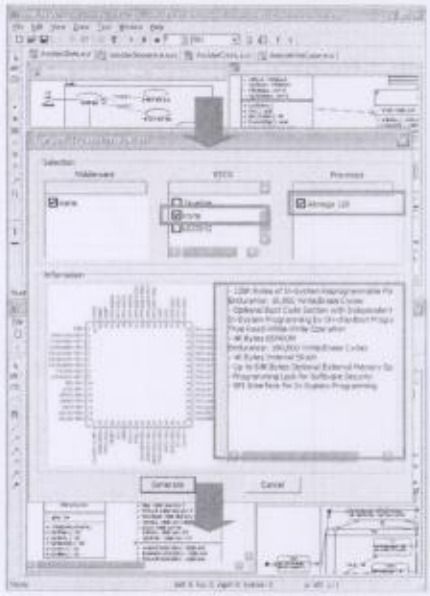


Figure 12. Transformation T1 : Multi-Joint Robot 1

Figure 13 shows to click 'none' button on Middleware, choose Javeline on RTOS, and also Ubicom SX48AC on processor for Multi-Joint Robots 2. Then also click 'generate' button to automatically generate other TSM.

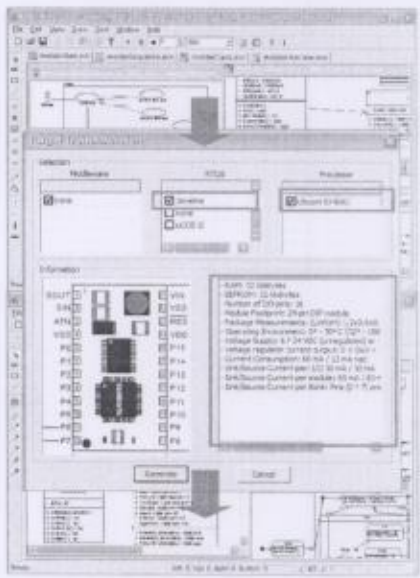


Figure 13. Transformation T1 : Multi-Joint Robot 2

C. *TDC(Target dependent code) Phase*

The TDC phase is automatically generated the target dependent code per each different target system. Its possible generable code languages are C, C++, and Java.

Table II shows to compare the automatically generated codes of heterogeneous multi-joint robots through the tool.

TABLE II. THE GENERATED HETEROGENEOUS CODE WITH OUR AUTOMATIC TOOL

(a) Multi-Joint Robots 1	(b) Multi-Joint Robots 2

#define THIS m_ServerMotor;
typedef struct tagServoMotor
{
struct Arm super;
void (*PluseOut)(int value);
}ServoMotor;
 	
void ServoMotor _Init(void *_this)
{
ServoMotor *THIS;
THIS = (ServoMotor *)_this;
THIS->PluseOut=
ServoMotor_PulseOut;
}
 	
void ServoMotor_PulseOut(void *_this , int values)
{
ServoMotor *THIS;
THIS = (ServoMotor *)_this;
pluse (THIS->super.m_speed, THIS->super.m_pin);
}
	class ServoMotor extends Arm
	{
	
	public ServoMotor()
	{
	}
	
	public void PulseOut(int values)
	{
	CPU.pluseOut(m_speed,m_pin);
	}
	
	}

V. CONCLUSION

It may be hard to reuse embedded software, such as model, design, and code, for the hardware dependent systems, that is, heterogeneous multi-joint robots.

We propose the semi-automatic software development based on MDD (Model Driven Development) to develop the heterogeneous systems. To solve a problem for the heterogeneous embedded software development, we implement the automatic tools for model transformation (Such as TIM and TSM) and code generation.

As a result, this leads to reduce the lifecycle of the heterogeneous embedded software development. We are still researching on the extending profiles and the verification tool.

ACKNOWLEDGEMENT

This research is financially supported by the Ministry of Education, Science Technology (MEST) and Korea Industrial Technology Foundation (KOTEF) through the Human Resource Training Project for Regional Innovation(2008~2009).

REFERENCES

[1] Daniel H. Wilson, How to Build a Robot Army, Bloomsbury, 2008

[2] A. Kleppe, J.Warmer, W.Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wiseley, 2003.

[3] Object Management Group, OMG Unified Modeling Language Specification (draft) Version 1.3, June 1999.

[4] Leon Starr, Executable UML: How to Build Class Model, Prentice Hall, 2002.

[5] Mellor, Stephen J., Marc J.Balcer, Executable UML: A Foundation for Model-Driven Architecture. Boston: Addison-Wesley, 2002.

[6] Bart Broekman, Edwin Notenboom, Testing Embedded Software, Addison-Wesley, 2003.

[7] Pierre Boulet, Jean-Luc Dekeyser, Cedric Dumoulin, and Philippe Marquet, "Mda for Soc Design, Intensive Signal Processing Experiment", In FDL'03, Frankfurt, September 2003. ECSI.

[8] D. Kim, W. Kim, Robert Y. Kim, "A Study on Design for Embedded S/W based on Model Driven Architecture", Journal of IWIT, Korea, Vol. 6, No. 1, March 2006.

[9] Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe, "Feature-Oriented Product Line Engineering," IEEE Software, Vol. 9, No. 4, Jul./Aug. 2002, pp.58-65.

[10] Jean-Louis Houberdon, Jean-Philippe Babau, "MDA for embedded systems dedicated to process control," Workshop on MDA in SIVOEES, in conjunction with UML'2003, October 2003.

[11] W. Kim, Robert Y. Kim, "A Study on Extension of Executable UML for Modeling Real-time Embedded Software", Proceedings of the 25th KIPS Spring Conference, Korea, Vol. 13, No. 1, May 2006, pp.231-234.

[12] W. Kim, H. Son, Y. Park, B. Park, C. Carlson, R. Kim, "The Automatic MDA (Model Driven Architecture)Transformations for Heterogeneous Embedded Systems", SERP08, July 2008.

[13] W. Kim, Robert Y. Kim, "A Study on Modeling Heterogeneous Embedded S/W Components based on Model Driven Architecture with Extended xUML", The KIPS Trans., Vol. 14-D, No. 1, Feb. 2007, pp. 83-88.

IEEE Computer Society Order Number E3546
BMS Part Number CFP0882F-CDR
ISBN 978-0-7695-3546-3

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Silvia Ceballos
Cover art production by Mark Bartosik

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps