# Integrated Environment of Simulation and Real-Time Control Experiment for Control system

Kentaro Yano and Masanobu Koga
*Kyushu Institute of Technology*
*Japan*

## 1. Introduction

A design process of a control system is generally executed in order of modelling, design of controller, simulation, and control experiment. If a control plant is a robot or an inverted pendulum etc, Real-Time control is required and control experiment programs should be Real-Time programs (RT programs). An RT program is the program which assures the time limit of the process beginning and the process completion (Funaki & Ra, 1999). And, the control experiment program which is an RT program is called an RT control program.
An RT control program is often written by using a library provided by a Real-Time OS (RTOS) like RT-Linux (RTLinuxFree; Funaki & Ra, 1999). At this time, it is necessary to find the parts which should be altered by the change of the control plant from whole of the program. Since, the target-depend parts are scattering at large range of the program. Also, there is a high possibility that the miss which forget the partial change etc. get mixed in with the program.
A simulation is run to confirm the performance of the controller, and a simulation program is often written in a numerical computation language which makes it easy to write a mathematical formula (The MathWorks Matlab; Koga, 2000). After the affirmation of the results of the simulation, an RT control program is newly written. Therefore, it is impossible to execute the design process of control system efficiently, because an individually creation of a simulation program and an RT control program is needed and smoothly change from simulation phase to control experiment phase is impossible.
To solve this issue, methods which create RT control programs using the information written at simulation programs are proposed. For example, by using RtMaTX (Koga et al., 1998), it is able to create RT control programs by edit the function written in MaTX (Koga, 2000) which is a numerical computation language. And, Real-Time Workshop (RTW) (The MathWorks Real-Time Workshop) generates RT control programs written in C language from block diagrams of Simulink (The MathWorks Simulink). A method which improves RTW to industrial applications and generates iFix (GE Fanuc Automation) etc. from Matlab codes is also proposed (Grega & olek, 2002).

A simulation is run on a common OS like Windows, but an RT control is often run on an RTOS. And, the platform (hardware or OS) for a simulation and the platform for an experiment are often different. Then, it is necessary to deploy the RT control program which is automatically generated from a simulation program on the machine which executes a control experiment. Since a simulation and a control experiment are executed repeatedly, it is necessary to repeatedly change a simulation program, create an RT control program, and deploy it on an experiment machine.
To reduce a workload of a deployment of an RT control program, a method which uses a Web browser is proposed (Basso & Bangi, 2004). In this method, users can execute an automatic generation of an RT control program and control experiment by only selecting the simulation program written in Simulink on a Web browser. However, the simulation is executed by Simulink, and the RT control is run by the Web browser, so because of the difference of the interface, users are easy to be confused by the operations. And, HTML client which is the application that runs on a Web browser and uses UI parts provided by HTML, has problems like it is impossible to provide rich functions because the UI is poorer than an UI of a common program.
The purpose of this paper is to propose methods to solve the issues and make it possible to efficiently execute the iteration of the simulation and the Real-Time control at the design process of control system. And, this paper describes about the integrated environment for simulation and Real-Time control which is the implementation of the proposed methods.
In Section 2, we propose the methods for the issues. In Section 3, the developed integrated environment to actualize proposed methods is described. In Section 4, an illustrative example which is a stabilization control of inverted pendulum is explained. Finally, in section 5, conclusions are described.

## 2. Integrate methods of simulation and Real-Time control

### 2.1 RT Control Framework

In this section, we propose an RT control framework which provides a frame of the creation of RT control programs for target-dependent parts are scattering at the large range of an RT program.

#### 2.1.1 Framework

In software engineering, a framework is used commonly in the sense of the technique which provides a frame of the whole software (Nakano, 2002). Based on a framework, it is possible to complete a final product efficiently by the creation where a developer corresponds to individual purposes. So, a framework is provided as a semi finished product. The parts which are changed or added by developers are called hotspots, and other parts are called frozen spots.
As shown in Fig. 1, a program written by users using a library only calls and uses codes in a library. On the other hand, in a framework, it calls and uses codes developed by users based on a specification of a framework.

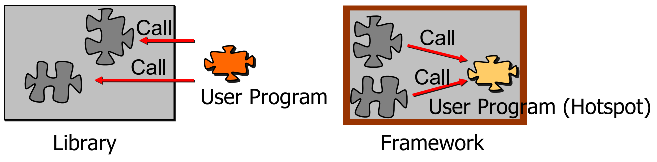


Fig. 1. Framework and Library

### 2.1.2 Architecture

The architecture of the framework which we propose is shown in Fig. 2. The RT control framework provides the frame of the creation of RT control programs. Target-dependent parts become apparent and it is able to control centrally by using the RT control framework. It is also possible to create an RT control program by only creating the target-dependent parts. Therefore, it makes writing RT control programs easy, and it is possible to raise the efficiency. It is possible to write an RT control program only creating pieces of Real-Time task, experimental data viewer, and commands interpretation. The details of the architecture are shown below.

- Real-Time Task module

Handle Real-Time processing, and compute the control law etc.

- Real-Time Task Management module

Generate/management the Real-Time Task

- Experimental Data Management module

Manage the experimental data

- Experimental Data Viewer module

Display the experimental data

- User Interface module

Handle commands (start/end of control etc.) from the user

- Commands Interpretation module

Interpret commands from the user, and handle them.

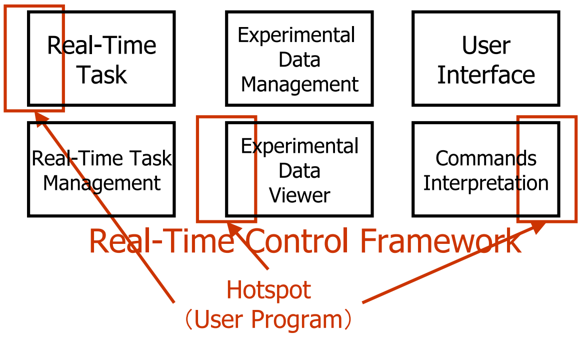


Fig. 2. Architecture of framework

## 2.2 Transformation of simulation program using object model

To solve the issues about simulation programs depend a platform, and individually creation of a simulation program and an RT control program is needed, this paper proposes the transformation of a simulation program using the object model and Design Pattern (Gamma et al., 1995). First, this paper proposes a method which transforms a simulation program written in a numerical computation language to a simulation program written in a platform independent language. And, this paper also proposes a method which generates a Real-Time processing code（hotspots of the RT control framework）using Factory Method pattern (Gamma et al., 1995). It is able to raise the efficiency of the design process of control system by generating an RT control program automatically from a simulation program written in a numerical computation language. It is able to restrict the generating program to target-dependent parts by generating the hot spots of the RT control framework. And, it is able to execute a simulation and a control experiment on the same platform by transforming a simulation program to a platform independent language.

### 2.2.1 Object Model

An object model is the data structure which is the object oriented model of the data used by a program, and it is the interface of given operations for that. An object model of a programming language is a set of classes to create the objects which have same information of the source code that was written in that language. Figure 3 shows the schematic of the object modeling of a source code as an object tree which has the same information of the source code. Let a function which generates another programming language is defined at each node. Then, it is possible to transform the original source code to another programming language by pursuing the object tree and calling the defined functions written at each node.

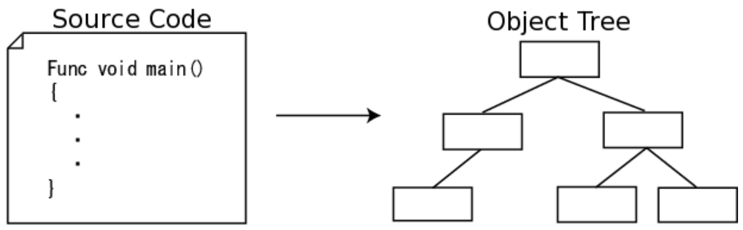


Fig. 3. Object Model

### 2.2.2 Transformation of program using object model

Figure 4 shows the schematic of the transformation of a simulation program using the object model. First, Parser receives a source code written in a numerical computation language.
Parser construes the source code, and creates the object tree which has the information of the source code written in the numerical computation language based on the syntax. Objects which define a function to generate a platform independent code are used to build the tree. Then, the simulation program written in the platform independent language is generated from the generated object tree.

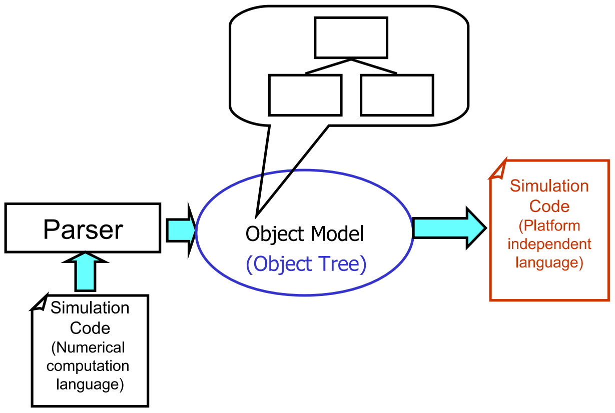

Parser
Object Model
(Object Tree)
Simulation Code (Platform independent language)
Simulation Code (Numerical computation language)

Fig. 4. Transformation of simulation program using Object Model

### 2.2.3 Generation of RT program using Factory Method pattern

Factory Method pattern is a design pattern which decides how to create an instance of a class defined at a super class, and the concrete creation of an instance is handled by each subclass (Gamma et al., 1995). The Factory Method Pattern defines that Creator creates Product. And ConcreteProduct which is the real instance is created by ConcreteCreator which is the subclass of Creator. Then, the class which provides the frame of the instantiation and classes for the generation of the real instance are separated.
It is possible to switch objects which are embedded in an object tree for any purpose by using Factory Method pattern when creating an object tree.
Figure 5 shows the schematic of the transformation of a simulation program using Factory Method pattern. Factory Method pattern are applied where Parser creates the object model. And, it defines that Creator creates the object model (Product). Objects (ConcreteProduct) which is used to generate a simulation code is created by SimCreator (ConcreteCreator) which is an implementation class of Creator.
Figure 6 shows the schematic of the automatic generation of an RT program using Factory Method pattern. Objects（ConcreteProduct）which generate an RT program automatically is created by RTCreator(ConcreteCreator）which is a subclass of SimCreator. Then, it is able to transform a platform independent simulation program and generate a Real-Time processing code automatically by using the object model which has the same structure. If a subclass of Creator and classes which are embedded in the object tree are created, it is possible to add another kind of code.

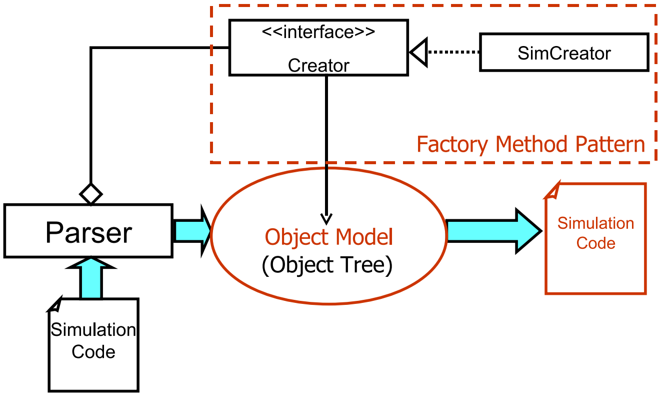


Fig. 5. Transformation of simulation program using Factory Method Pattern

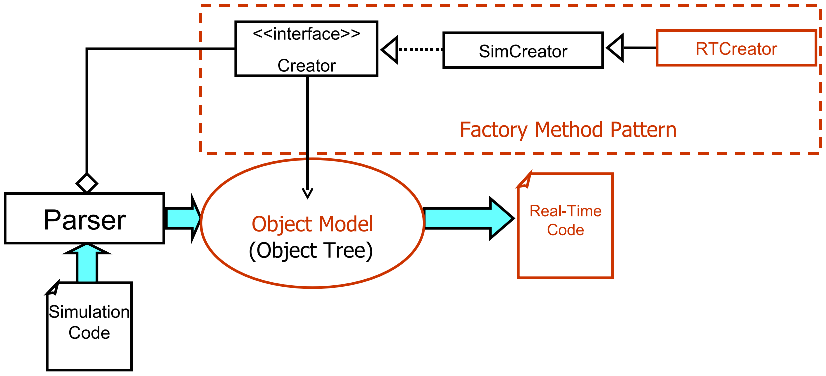


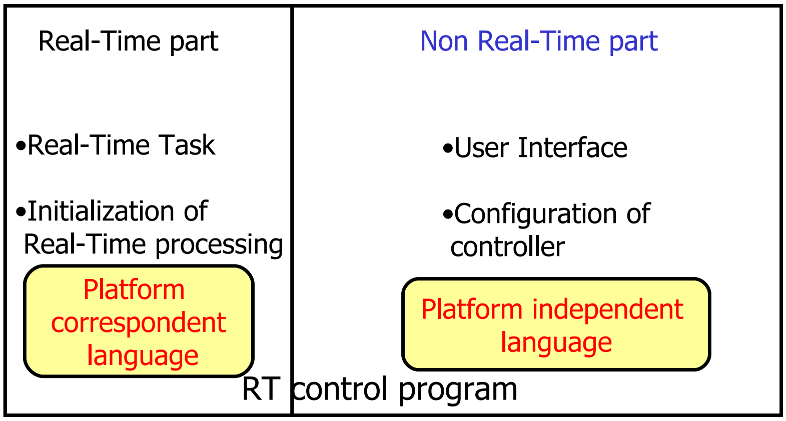
Fig. 6. Automatic generation of Real-Time control program using Factory Method Pattern

### 2.3 Separation of platform dependent parts

To solve the issue of RT control programs depend on a platform, this paper proposes a method which separates platform dependent parts and independent parts. And, this paper proposes that separates platform dependent parts of the RT control framework.

#### 2.3.1 Separation of platform dependent parts of RT control program

In general, an RT control program consists two parts. One is Real-Time part which needs a Real-Time processing, and another part is non Real-Time part which doesn't need a Real-Time processing. If Real-Time part is written in a platform correspond language, and non Real-Time part is written in a platform independent language, non Real-Time part can be independent from a platform. The outline about the separation of platform dependent parts is shown by Fig. 7.


Real-Time part
Non Real-Time part
•Real-Time Task
•Initialization of Real-Time processing
•User Interface
•Configuration of controller
Platform correspondent language
Platform independent language
RT control program
Fig. 7. Separation of platform dependent parts

**2.3.2 Server and client system**

This section describes an improvement of a portability of programs by combining of the method mentioned the above section and server and client system which use network.
Figure 8 shows the schematic of server and client system. A plant is connected to a server machine. The server machine is installed an RTOS, and it can execute RT control programs. The client machine is connected to the server machine via network.
In this system, Real-Time processing is executed by only the server machine, and the client machine runs only non Real-Time processing. And, non Real-Time information is sent between the server machine and the client machine. So, the client machine doesn't need the special environment for Real-Time programs. And, the client machine doesn't depend on the particular OS, since non Real-Time processing parts are platform independent. So, the combining of the separation of platform independent parts and server and client system improve the portability of the program.
And, if a mechanism which automatically send an RT control program from the client machine to the server machine is implemented, the issue about deploy of an RT program is solved.

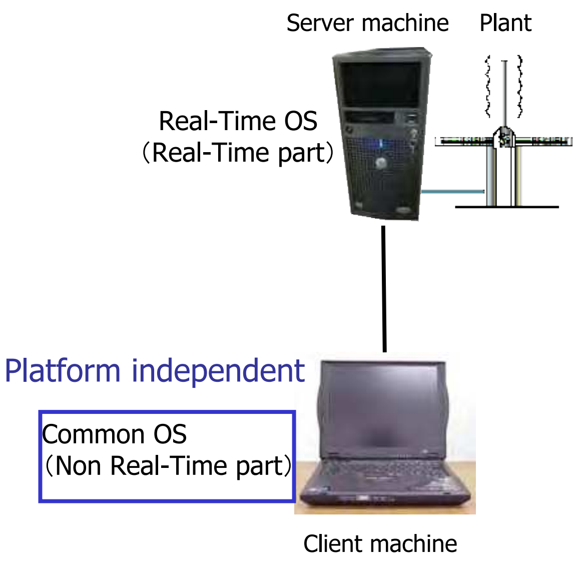


Fig. 8. Server and client system

## 3. Implementation of proposed method

We implemented the proposed methods, and developed the integrated environment for simulation and Real-Time control experiment using C language and Java language.
Figure 9 shows the schematic of the integrated environment.
An RT control program using the RT control framework is generated automatically from a simulation program by the integrated environment. The integrated environment can raise the efficiency of the design process of control system, since users can run the integrated simulation and Real-Time control experiment.

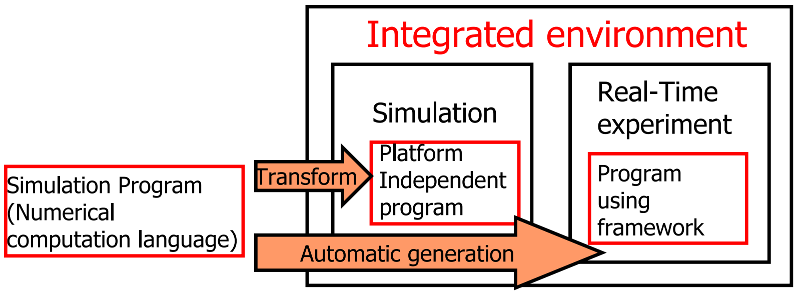


Fig. 9. Schematic of Integrated Environment

Figure 10 shows the architecture of the developed integrated environment. The RT control framework runs on an RTOS and the Java VM. The framework has two hotspots. Hotspot 1use the function of an RTOS, and hotspots 2use the function of the Java VM. It is able to execute the simulation and the control experiment by using the GUI which runs on the framework. And the automatic code generation is used for the simulation and the control experiment.

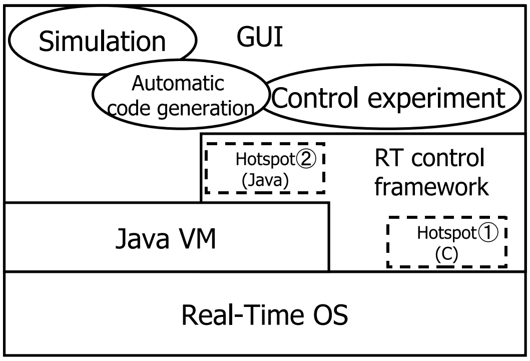


Fig. 10. Architecture of Integrated Environment

### 3.1 RT control framework

We developed ReTiCoF(Real Time Control Framework) (Yano & Koga, 2006) which is the implementation of the Real-Time control framework proposed by this paper. ReTiCoF runs on RT-Linux (RTLinuxFree) which is one of the RTOS, and it is implemented by C language and Java language.
The architecture of MK-Task which is the execution parts of Real-Time processing in ReTiCoF is shown in Fig. 11. MK-Task is implemented by C language, and it provides mechanisms for easy use of functions of RT-Linux.

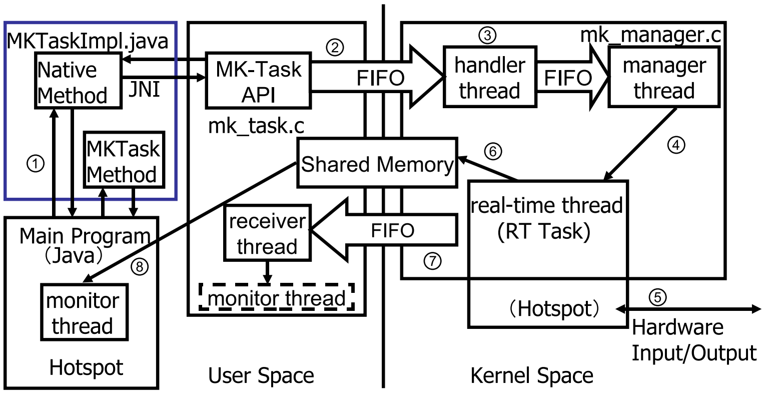


Fig. 11. Architecture of MK-Task

MK-Task consists of mk_task.c which runs on User space, and mk_manager.c which runs on Kernel space. mk_task.c has the APIs of MK-Task, and codes of receiver thread which receives data from Kernel space. mk_manager.c has the handler thread which runs when a call from User space is sent, manager thread which manages a Real-Time task, and Real-Time task as thread which has the Real-Time constraint. And MKTaskImpl.java has native

methods which call the APIs of MK-Task using JNI, and methods which are needed when MK-Task is used from Java. JNI(Java Native Interface) (Gordon, 1998) is the technique to use a native code like C language from Java.
The process flow of the Real-Time control using MK-Task is shown below.

1. Main program(Java) calls the function of MK-Task as a native method of MKTaskImpl class
2. A request form user space using the function of MK-Task is sent to Kernel space via RT-FIFO
3. Handler thread starts
4. The request received by handler thread is passed to manager thread, and manager thread manages the RT task as required
5. The RT task inputs or outputs to a hardware, and execute the Real-Time control
6. Experimental data of the Real-Time control are written into the shared memory of user space and kernel space
7. Receiver thread receives total count of data from RT task via RT-FIFO
8. Monitor thread receives the data from the shared memory, and display the data

### 3.2 Transformation of program using object model

The developed integrated environment employed MaTX as the numerical computation language, and Java as the platform independent language. The integrated environment creates an object model of MaTX from a source code of MaTX using matj(Koga et al., 2005) which is an interpreter of MaTX implemented by Java, and generates a source code of Java by pursing the object model.
Factory Method pattern is applied to the Parser of matj, and a Real-Time processing code (hotspots of the RT control framework) is generated automatically from a simulation program written in MaTX. In particular, we created classes which generates Real-Time code(matrix calculations, trigonometric functions etc.), and a class which embeds those classes to an object tree.
The integrated environment is also able to generate an RT control program from a block diagram created by Jamox (Koga et al., 2006). The block diagram which is supported by the integrated environment has a controller created by a block of MaTX or cascade connections of linear systems.
Jamox can run the modeling and the simulation, so it is possible to execute all parts of the design process of control system by using Jamox and the integrated environment.
Jamox(Java Agile MOdeling Tool for Control System) is a modeling and simulation tool for control system using an adjacency matrix, and it supports a linear system and a nonlinear system modeled by continuous time, discrete time, and sampled data system(which is a mix of continuous time and discrete time).
An adjacency matrix is one of an expressive form of a graph used for graph theory. It is easy to simulate a control system represented by a block diagram drawn by Jamox. Jamox not only can simulate time response of system, but also can compute a state space representation of all system and frequency response (Bode diagram) easily, if that system is a linear.
Figure 12 shows an execution screen of Jamox. It is able to allocate blocks (provided by the tool) by drag and drop from the library to the canvas. Each block is able to be connected by mouse action, and it is easy to create a block diagram on the screen displays.

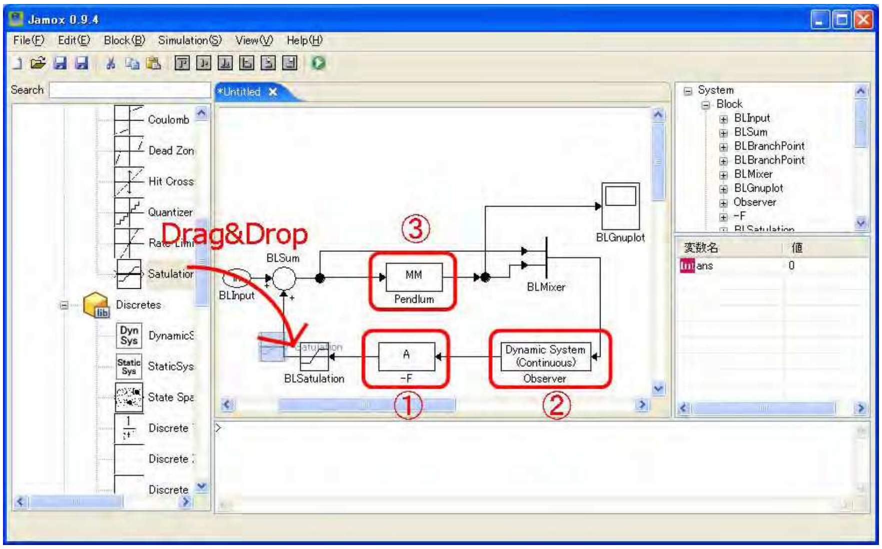


Fig. 12. Modeling and Simulation tool Jamox

The block 1is a basic block registered by the library. Kinds or the structure of the models are defined, and only the parameter is customizable. The block 2is a customer-defined block which represents any dynamic systems and static systems. And it is related with the Java class which extends *DynamicSystem* class or *StaticSystem* class. It is able to create the high reusable blocks efficiently based on the grouping of the control system and use of the inheritance of class.And, it is easy to create the fast computation code using NFC (Koga & Matsuki, 2003) which is a Java numerical computation package. The block 3is a block which treats a system written in MaTX syntax, and it is possible to reuse existing MaTX codes.
In the simulation calculation, the solution of a differential equation is solved by *RungeKutta* class etc. which implements *OdeSolver* interface provided by Ode package of NFC. If an algebraic loop exists, a solution of a nonlinear simultaneous equation is solved by using Newton-Raphson method.

### 3.3 Separation of platform dependent parts

We use C language for the platform correspond language and Java language for the platform independent language. We use JNI to use C language and Java language together.
In this integrated environment, server and client system is implemented by using RMI(Remote Method Invocation) (Trail: RMI) which is the implementation of distributed objects in Java. The distributed object is an object which is able to call methods through network. And, we implement the mechanism which sends an RT control program from a client machine to a server machine using RMI.

### 3.4 Local and remote compatible GUI

Figure 13 shows the GUI provided by the integrated environment. It has the menu bar and the toolbar which are related to the simulation, the automatic generation, and the control experiment. It also has the display part of the experimental data, the property of the Real-Time task, and messages for user.

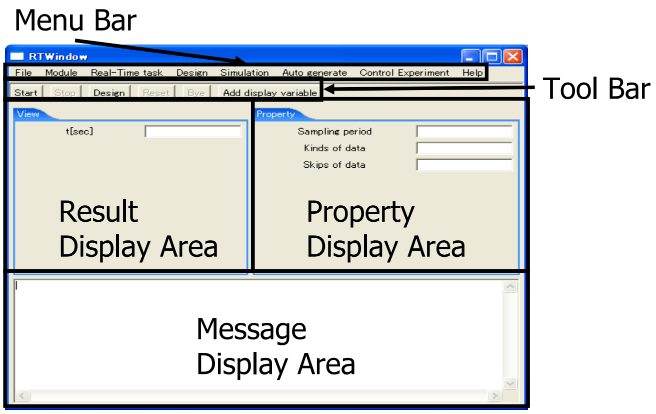


Fig. 13. GUI of integrated environment

In this implemented integrated environment, it is possible to use the same GUI between the local control using only the server machine, and the remote control using the server machine and the client machine. Proxy pattern (Gamma et al., 1995) is used for the mechanism of the local and remote compatible GUI. Proxy pattern is the design pattern which makes Proxy class as the proxy of RealSubject class which is the real actor, and it handles processes of RealSubject class as much as possible.
Figure 14 shows the class diagram of the integrated environment using Proxy pattern. The GUI which is shown by Fig. 13 is provided by RTWindow class. RTWindow class has the classes which provide the GUI parts like ViewPanel class, and the implementation class of MKTask interface which executes the Real-Time control. MKTaskImpl class is used for the local control, RemoteMKTaskProxy(Proxy) class is used for the remote control instead of RemoteMKTaskImpl(RealSubject).

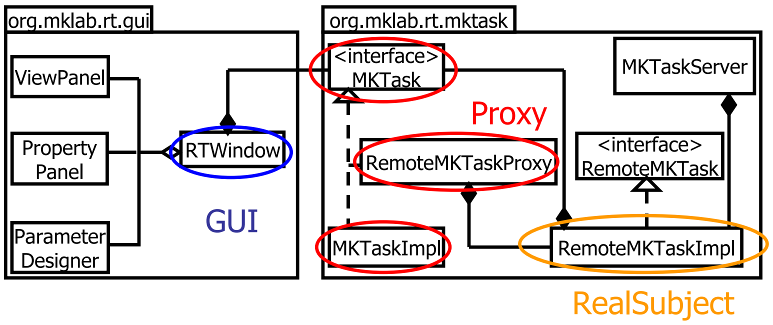


Fig. 14. Class Diagram of Real-Time control system

## 4. Experiment of stabilization control of an inverted pendulum

To verify the effectivity of the method proposed by this paper, we execute the experiment of the stabilization control of an inverted pendulum using the developed integrated environment.
First, do modeling a pendulum using Jamox and design a controller using MaTX. Figure 15 shows the result of the modeling and the design of controller.

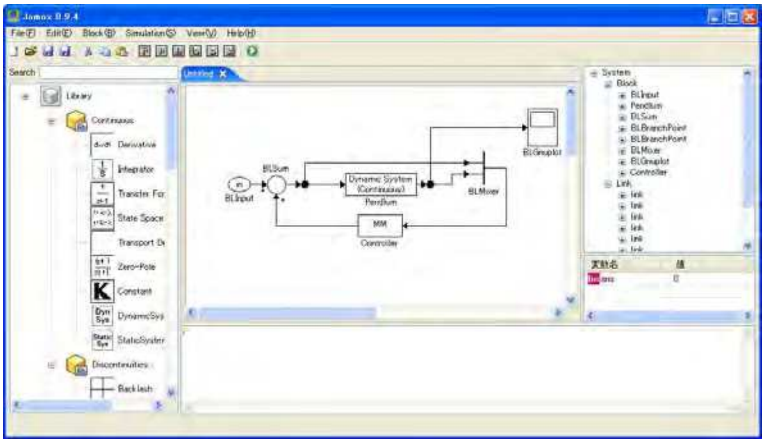


Fig. 15. Modeling of pendulum using Jamox

Simulation is executed by using Jamox. At this time, a user selects a solver of the differential equation, and input the start/end time of the simulation, and the initial state. Figure 16 shows the results of the simulation.

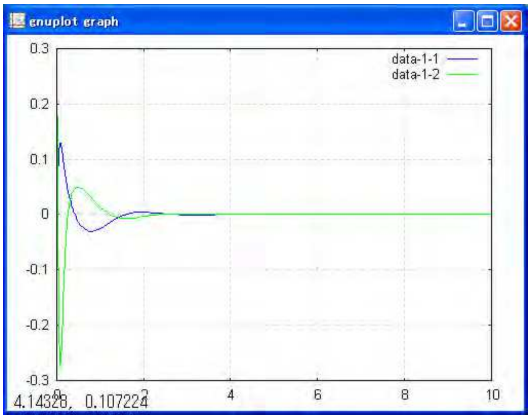
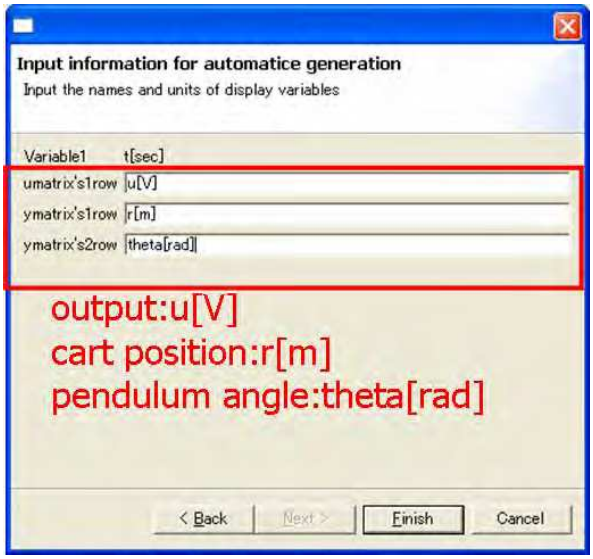


Fig. 16. Execution of simulation

In the automatic generation of Real-Time program, user selects the simulation program(created by using Jamox), the function name to extract information, the name of the

output variable, and input the name/unit of display variables for the GUI. Figure 17 shows the dialog provided by the GUI of the integrated environment, and it is used to input the name/unit of display variables for the GUI.
After the above information is inputted, the Real-Time program is generated automatically. And the variable displayed on the GUI is added, the display of the GUI is changed for the pendulum.

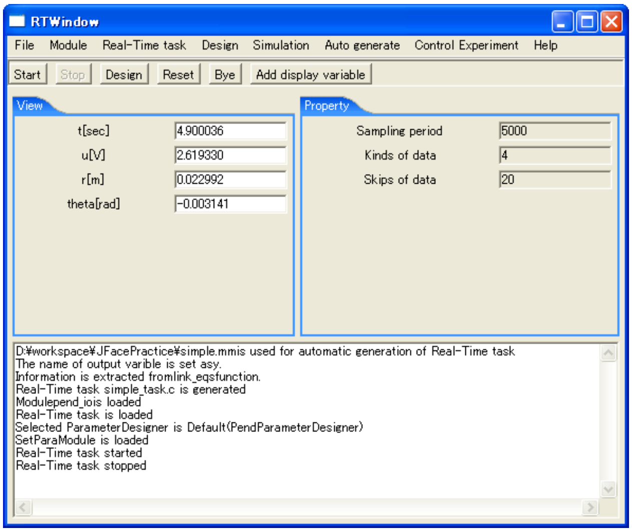

Input information for automatice generation
Input the names and units of display variables

Variable1 t[sec]
umatrix's1row u[V]
ymatrix's1row r[m]
ymatrix's2row theta[rad]

output:u[V]
cart position:r[m]
pendulum angle:theta[rad]

Fig. 17. Display variable name・unit for GUI

The experiment is executed, after select the generated RT control program, the module of the experimental apparatus, and input the parameter of the controller. Figure 18 shows the screen of the RTWindow which is executing the control experiment.
It is shown that the GUI has changed for the inverted pendulum by using the information input by Fig. 17 by comparison Fig. 13 and Fig. 18.
Figure 19 shows the experiment results, and abscissa axis is time[sec], ordinate axis is angle of the pendulum[rad]. And, the sampling time of the experiment is 5[ms].
As shown in Fig. 19, the angle of the pendulum is close to the 0[rad], the experiment of the stabilization control of the inverted pendulum become successful.



| | |
|---|---|
| t[sec] | 4.900036 |
| u[V] | 2.619330 |
| r[m] | 0.022992 |
| theta[rad] | -0.003141 |

| | |
|---|---|
| Sampling period | 5000 |
| Kinds of data | 4 |
| Skips of data | 20 |

```
D:¥workspace¥JFacePractice¥simple.mmis used for automatic generation of Real-Time task
The name of output varible is set asy.
Information is extracted fromlink_eqsfunction.
Real-Time task simple_task.c is generated
Modulepend_iois loaded
Real-Time task is loaded
Selected ParameterDesigner is Default(PendParameterDesigner)
SetParaModule is loaded
Real-Time task started
Real-Time task stopped
```

Fig. 18. Execution of control experiment

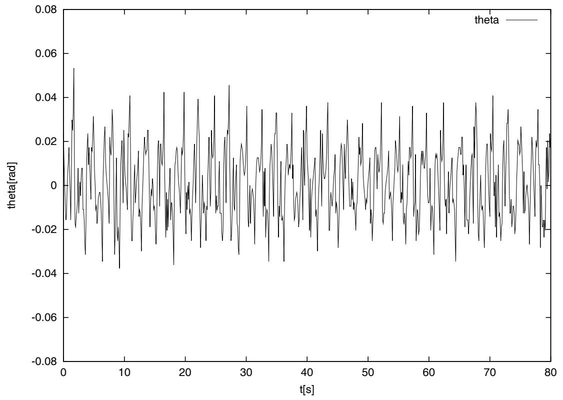


Fig. 19. Experiment results of stabilization of inverted pendulum

## 5. Conclusions

This paper proposed the methods which make the execution of the iterative design process of control system efficiently. In particular, this paper proposed the method which is based on the RT control framework, transformation of a program using the object model, and separation of platform dependent parts. And, we developed the integrated environment for the simulation and the Real-Time control experiment which is the implementation of proposed methods. The effectivity of the proposed methods was shown by the stabilization of an inverted pendulum.

In the implementation example, we used RT-Linux as the RTOS, but a method which runs RT control program on Linux Kernel 2.6(Kishida & Koga 2005) is proposed. So, we would like to make the integrated environment is corresponded to Linux Kernel 2.6 in future works.

## 6. References

Basso, M. & Bangi, G. (2004). ARTIST:A Real-Time Interactive Simulinkbased Telelab, *proceedings of the 2004 IEEE Conference on Computer Aided Control Systems Design*, pp.196-201

Funaki, M. & Ra, S. (1999). *Guide book for Real-Time sensing and control by Linux*, Shuwa system, 97804879668493

Gamma, E.; Helm, R. Johnson, R. & Vlissides, J. (1995). *Design Patterns:Elements of Reusable Object-Oriented Software*, Addison-Wesley Pub, 978-0201633610

GE Fanuc Automation: Proficy HMI/SCADA - iFIX, http://www.gefanuc.com/en/ProductServices/AutomationSoftware/Hmi Scada/iFIX/

Gordon, R. (1998). *Essential JNI: Java Native Interface*, Prentice Hall Ptr, 978-0136798958

GREGA, W. & KOLEK, K. (2002). Simulation and Real-Time Control:from Simulink to Industrial Applications, *2002 IEEE International Symposium on Computer Aided Control System Design Proceedings*, pp.104-109

Kishida, K.; Koga, M. (2005). Development of Real-Time Control Package with Linux Kernel 2.6, 49th Annual Conference of the Institute of Systems, Control and Information Engineers

Koga, M.; Tsutsui, Y. & Yabuuchi, J. (2006). Java Simulation Platform for Control System based on Block Diagram, *IEEE 2006 CCA/CACSD*, pp.2304–2308

Koga, M.; Matsuki, T & Sada, H. (2005). Development of Environment of Numerical Computation in Java based on Object Model of Programming Language, 49th Annual Conference of the Institute of Systems, Control and Information Engineers

Koga, M. & Matsuki, T. (2003). Development of OS-Neutral Numerical Foundation Class Library and its Application to Control System, *SICE 6th Annual Conference on Control Systems*

Koga, M. (2000). *MaTX for control/numerical analysis*, Tokyo Denki University Press, 978-4501531003

Koga, M. Toriumi, H. & Sampei, M. (1998). An integrated software environment for the design and real-time implementation of control systems, *Control Engineering Practice*, Vol. 6, pp. 1287–1293

Nakano, T. (2002). Understanding the framework, *Java WORLD*, Vol. 6, No. 62, 54–67

RTLinuxFree, http://www.rtlinuxfree.com/

The MathWorks Matlab, http://www.mathworks.com/products/matlab/

The MathWorks Real-Time Workshop, http://www.mathworks.com/products/rtw/

The MathWorks Simulink, http://www.mathworks.com/products/simulink/

Trail: RMI http://java.sun.com/docs/books/tutorial/rmi/index.html

Yano, K. & Koga, M. (2006). Platform Independent Integrated Environment for Simulation and Real-Time Control Experiment, *SICE-ICASE International Joint Conference 2006*

**Mechatronic Systems Simulation Modeling and Control**
Edited by Annalisa Milella Donato Di Paola and Grazia Cicirelli

ISBN 978-953-307-041-4
Hard cover, 298 pages
**Publisher** InTech
**Published online** 01, March, 2010
**Published in print edition** March, 2010

This book collects fifteen relevant papers in the field of mechatronic systems. Mechatronics, the synergistic blend of mechanics, electronics, and computer science, integrates the best design practices with the most advanced technologies to realize high-quality products, guaranteeing at the same time a substantial reduction in development time and cost. Topics covered in this book include simulation, modelling and control of electromechanical machines, machine components, and mechatronic vehicles. New software tools, integrated development environments, and systematic design methods are also introduced. The editors are extremely grateful to all the authors for their valuable contributions. The book begins with eight chapters related to modelling and control of electromechanical machines and machine components. Chapter 9 presents a nonlinear model for the control of a three-DOF helicopter. A helicopter model and a control method of the model are also presented and validated experimentally in Chapter 10. Chapter 11 introduces a planar laboratory testbed for the simulation of autonomous proximity manoeuvres of a uniquely control actuator configured spacecraft. Integrated methods of simulation and Real-Time control aiming at improving the efficiency of an iterative design process of control systems are presented in Chapter 12. Reliability analysis methods for an embedded Open Source Software (OSS) are discussed in Chapter 13. A new specification technique for the conceptual design of self-optimizing mechatronic systems is presented in Chapter 14. Chapter 15 provides a general overview of design specificities including mechanical and control considerations for micro-mechatronic structures. It also presents an example of a new optimal synthesis method to design topology and associated robust control methodologies for monolithic compliant microstructures.

**How to reference**
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Kentaro Yano and Masanobu Koga (2010). Integrated Environment of Simulation and Real-Time Control Experiment for Control system, Mechatronic Systems Simulation Modeling and Control, Annalisa Milella Donato Di Paola and Grazia Cicirelli (Ed.), ISBN: 978-953-307-041-4, InTech, Available from: http://www.intechopen.com/books/mechatronic-systems-simulation-modeling-and-control/integrated-environment-of-simulation-and-real-time-control-experiment-for-control-system