A HUMAN MODEL USING COMBINED FUNCTIONS OF UML, MATLAB/SIMULINK AND CODE-LIBRARY

1,2Arai, Masatoshi*; 3Eguchi, Toru; 1Yoshida, Norihiko

1Saitama University, Japan; 2Calsonic Kansei Corp., Japan; 3FUJITSU COMPUTER TECHNOLOGIES LIMITED, Japan

KEYWORDS:
MATLAB/Simulink; UML; Human model; Multiple-task; Physical-control model;

ABSTRACT.
This paper proposes a new framework for a human model based on a V-process following ISO12207. For the purpose of easy design and test, physical models and code-libraries are integrated in UML (Unified Modelling Language) as components. Maintainability and reusability of the components can be improved when integrated in UML. It is because all the individual parts are functionally integrated, and their relationships are presented explicitly in UML. This is the first time that the framework is proposed according to the V-process on which physical models and code-libraries were integrated using UML. Now-a-days, the demand for integration framework is increasing, as the human model is getting more complex.

1. INTRODUCTION

Recently, automatic operations to act human's driving are actively developed. A human driving model which modelled and built on an actual mechanism is very complex, because it contains many physical models and several device drivers written in code-libraries.

Physical models are generally designed in MATLAB/Simulink. Because MATLAB/Simulink has not only mathematical libraries of matrix operations but also functions of graphics characteristics. Therefore, it is easy for a designer to create and check functions of physical models.

On the other hand, device drivers of actual mechanisms are demanded for high response. So, these drivers are generally coded in C/C++ language and made a code-library.

The V-process is generally used in automotive developments. In this time, the V-process which is well known and is demanded a complex system like a human model. But there was no tool which treats physical models and code-libraries simultaneously in each process of the V-process.

A new framework which integrates MATLAB/Simulink models and code-libraries is made using an enhanced feature of UML on V-process.

There are two big challenges when a human model is constructed. One is how to combine MATLAB/Simulink models and code-libraries under a large-scale system based on the V-process. The other is to be able to describe multi-processing specification without special knowledge.

In this situation, various researches are proposed. For instance, researches that convert UML models and MATLAB/Simulink models have been proposed [1][2]. However, these researches do not describe how to handle code-libraries.

This paper proposes an integration using an enhanced combined UML function on V-process for a human model. The enhanced combined UML function consists of two new concepts. One is the “port” feature to combine MATLAB/Simulink model and code-libraries. The other is the multitasking feature to describe multi-processing specification easily. As a result, a large-scale system including a human modelling and its actual mechanism can be efficiently achieved.

The purpose of this paper using the V-process is described in section 2. Differences with other research are shown in section 3. Section 4 describes how to combine MATLAB/Simulink models and code-libraries. Section 5 shows an integrated experiment using UML based on the V-process. Finally, section 6 summarizes it.
2. PURPOSE OF THIS PAPER AND THE V-PROCESS

In recent years the V-process is widely used as a systems engineering process. An example of the V-process is shown in Figure 1. The V-process is derived from ISO 12207 standard. It has the main objective of supplying a common structure so that the buyers, suppliers, developers, maintainers, operators, managers and technicians involved in the software development use a common language.

In this paper, this proposal shows suitable for the V-process to develop very complex system like a human model.

The left side of the V-process is a design process or a modelling process, and the right side is a verification process. As for the left side, MATLAB/Simulink is generally used as a tool. The right side is a verification process of experimental mechanics using a real machine. As for the right side, device drivers for the real machine are coded in C/C++ language and registered as code-libraries.

![Figure 1 A V-process of the Systems Engineering Process](image)

Generally, a sequence on V-process flows from left side to right side. If a design on left side and a verification on right side in same level are executed simultaneously, its system development is very efficient. In order to flow the sequence horizontally and top down, a new framework using an enhanced UML is proposed.

Here, there are two important points,

1) UML can integrate systems easily by top down. The entire system of the V-process is integrated by UML.
2) The left side of V-process uses physical-models, i.e. MATLAB/Simulink models exist. It is necessary to combine MATLAB/Simulink models and code-libraries to execute the V-process horizontally.

3. COMPARISON BETWEEN OTHER RESEARCHES AND THIS PROPOSAL

To clarify differences with other researches, features of physical models: MATLAB/Simulink model and code-libraries are explained.

3.1 Physical Model: MATLAB/Simulink Model

A physical phenomenon is expressed by a differential equation, and modelled as what is called a physical model. An active damper physical model and its differential equations are shown in Figure 1. The active damper is applied to motors which rotate for themselves and prevents its resonance. This physical model is a model which suppresses the vibration of \( m_1 \) by choosing appropriate \( k_2 \) and \( c_2 \) for the damping of the vibrating mass \( m_1 \). This physical model can not only be used for a motor driving part of an electric vehicle but also is widely used as an actuator of a modelled human.
A physical phenomenon is manually represented by differential equations. If the differential equation can be formulated, it is possible to convert it into a MATLAB/Simulink model mechanically as shown in Figure 3. From Figure 2 and Figure 3, it is easy to understand an association of a differential equation and a MATLAB/Simulink model. MATLAB/Simulink models can represent physical models intuitionally. So, in this paper we define that MATLAB/Simulink models are equal to physical models.

A MATLAB/Simulink model (physical model) can simulate its function. Figure 3 on right shows a result of output of amplitude of \( x_1 \) when \( \sin(\omega t) \) signal is given as an input. From Figure 3, the output of amplitude of \( x_1 \) is made very small (the minus fifth power of ten). The result of output shows that the vibration has been lost.

A MATLAB/Simulink model can link to a mathematical library of matrix operations. MATLAB has both symbolic formula manipulations and the graphics facility. Using MATLAB, it is easy to generate a MATLAB/Simulink model and create a new function in the process of “Concept of Operation” of the V-process. From the equation of its model in Figure 2 and Figure 3, the suitable value of \( k_2 \) and \( c_2 \) are found and get the optimum amplitude magnification vibration graph as shown in Figure 4.

\[
\begin{align*}
\mu &= \frac{m_2}{m_1}, \nu = \frac{\omega_02}{\omega_01}, \lambda = \frac{\omega}{\omega_01}, \omega_{01} = \sqrt{\frac{k_1}{m_1}}, \omega_{02} = \sqrt{\frac{k_2}{m_2}}, \text{ and } \zeta = \frac{c_2}{2m_2\omega_{01}} \text{ are defined}
\end{align*}
\]
3.2 Code-library

A code-library is a software-parts library which is coded by C/C++ language and is reusable for another module. It is difficult to reproduce human actions. A human model is expressed separately for "plant" and "controller" as shown in Figure 5. A plant model is described by a physical model and it is controlled by an analogue signal. The plant needs digital-analogue converter and analogue operator. The output of the plant is read by a sensor and the sensor value is converted by analogue-digital converter. A digital-analogue converter, an analogue operator, a sensor, and analogue-digital converter are coded by C/C++ language and used by device drivers. These device drivers consist of for-looping and many if-then switches, so are written in C/C++. In this paper, we call these drivers as code-library.

Such large and complex system as a human model, needs physical models and code-libraries. This situation will be eternal in the future, and not going to change. Therefore, it is an important point how to combine these two.

3.3 Comparison with Other Researches

There is a very strong necessity for combing physical models with code-libraries as mentioned above. Various techniques are proposed in the world. The result of the comparison with this proposal and other proposals is shown in Table 1. Tamura[1] and Ozawa[2] have shown proposals that convert UML models and MATLAB/Simulink models. These two proposals only describe how to convert UML and MATLAB/Simulink models automatically, and did not explain how to combine MATLAB/Simulink models with code-libraries. According to their proposal, a MATLAB/Simulink model is converted into a new UML model automatically. So, it is difficult to understand and handle the result of an obtained UML model. On the other hand, using this proposal, an original MATLAB/Simulink model does not change its shape and is preserved.

MathWorks[3] doesn't have the function that connects with other systems. dSPACE[4] and Vector[5] do not have UML extensions, so, these products do not offer integrated management.
4. HOW TO COMBINE USING EXTENDED UML

An Integration of MATLAB/Simulink model and code-libraries under UML control is shown in Figure 6. There are two environments for development process, one is UML environment, and the other is MATLAB environment. UML environment is achieved by BricRobo[6] (FUJITSU COMPUTER TECHNOLOGIES LIMITED commercializes it) and operates on Enterprise Architect[7].

4.1 Two Environments

While considering a logical verification and its examination of a new concept, a physical model is operated only in MATLAB environment. When the physical model is combined with a code-library, the physical model is converted into C/C++ automatically. Users did not need to change the generated source code. Because a wrapper exists in UML environment and connects to the physical models. Therefore, an original physical model does not need to be changed. It is easy for user to treat physical models.

![BricRobo (UML) Environment](image)

**Figure 6** Overview for an Integration of Physical Model and Code-libraries under UML

4.2 Two New Extended UML Functions

UML environment has two new extended functions in order to combine and execute in parallel. One is an input output control function, and the other is multi-task control function.

1) Input output control function

   When combining MATLAB/Simulink models and code-libraries, it is necessary to send and receive data. Therefore, input port and output port are built on UML.

2) Multi-task control function
A MATLAB/Simulink model can execute its simulation by specifying interval time. In order to reproduce the simulation result on embedded software, function of the multi-task management is developed. Here, it should be necessary to specify the multi task easily.

4.3 How to Describe

An example of how to describe input output control and multi-task control is shown in Figure 7. Structure diagrams (composite structure diagrams) and behaviour diagrams (sequence diagrams) are used for the description. A structure diagram in Figure 7 has a product: “ProductA” consists of two parts: “P1” and “P2”. The “part” means an instance of MATLAB/Simulink models and code-libraries. It is necessary to specify an elapse time at the millisecond. Order of calling each part is written in Figure 7. The “P1:Part1” and the “P2:Part2” are called every 25 millisecond from the “Main-Task”. When we assumes that P1 is a code-library and P2 is a MATLAB/Simulink model, then the MATLAB/Simulink and the code-library are connected each other.

![Composite Structure Diagram and Sequence Diagram](image)

**Figure 7** An example of Composite Structure Diagram (left) and its Sequence Diagram (right)

5. INTEGRATED EXPERIMENT USING EXTENDED UML BASED ON V-PROCESS

It is shown to be able to apply this proposal according to the V-process horizontally. Following descriptions explain step by step corresponding to Figure 1. UML is used by two main purposes. One is make the V-process get detailed downward. The other is cooperation of horizontal works.

5.1 “Requirements and Architecture” and “System Verification and Validation”.

“Requirements and Architecture” step in Figure 1 gets a MATLAB/Simulink model from a physical phenomenon. It responds to an action of obtaining Figure 3 from Figure 2. It should not step forward without a verification of the MATLAB/Simulink model. When an actual plant exists: there is a motor or a human arm, it is understood that the MATLAB/Simulink model is correct from comparing the actual plant. An actual plant and MATLAB/Simulink model are located in parallel on UML, and a same input is given, a difference is output as shown in figure 8. It comes and goes horizontally in this V-process until the difference becomes small.

“System Identification Toolbox” in MATLAB function might use. This toolbox gets the input and the output data of an actual plant, and puts MATLAB/Simulink models directly.
5.2 “Detailed Design” and “Integration, Test, and Verification”.

“Detailed Design” is a controller design step to control the plant which is explained in section 5.1. When a controller is designed, it connects with a MATLAB/Simulink model and checks its function as shown in “Detailed Design” of the V-process. In “Integration, Test, and Verification”, the controller can test on an actual test. “Detailed Design” and “Integration, Test, and Verification” can be achieved by switching the connection of UML as shown in Figure 9.

5.3 Implementation

“Implementation” of bottom of Figure 1 has two actions, one is an auto code as shown in Figure 6. This converts MATLAB/Simulink models into C/C++ source codes. The other is building a target program on UML environment. This operation is “make makefile” and “compile” the target program. Use BricRobo environment, a makefile is autocoded.

6. CONCLUSION

This paper shows how to connect MATLAB/Simulink models and code-libraries. Two new extended functions are introduced and performed on the V-process process. It was a small trial at the model level of a human arm this time. However, using this approach has a possibility to be built a large system. It wants to be going to construct a large-scale human model next time, and to unite with the drive simulator.
REFERENCES


Arai, Masatoshi  Eguchi, Toru  Yoshida, Norihiko