Automatic Deployment of Heterogeneous Collaboration System Code with IMCL

Ju Li†, xxx†, xxx†, xxx†, xxx‡, xxx†*
† National Trusted Embedded Software Engineering Technology Research Center
East China Normal University, Shanghai, China
‡ LIPN and Paris University 7, Paris, France
Email: {jli, jwxiong, xmao, jqshi, xye, yhhuang}@sei.ecnu.edu.cn

***Abstract*—The abstract goes here.**

I. Introduction

This demo file is intended to serve as a “starter file” for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Preliminary

IMCL is an event-triggered language. The purpose of IMCL modeling language design is to model the actual industrial control domain system.

A. IMCL Mdodel

The modeling of complex industrial systems has different angles. The IMCL modeling method is based on the idea of refinement: modeling system functions, control logic, system resources, etc., layer by layer.

TODO: fig IMCL

- **System Layer** The system layer embodies the intuitive composition of a model. Usually, the system is composed of modular or functional components. For heterogeneous systems in the field of industrial control, controllers or processors with computational control capabilities within them can all run independently of each other. From the overall behavior of the system, the operation process is highly concurrent.
- **Scene Layer** The scenario layer describes the logical relationships between the independent components in the system, namely the control flow and interaction rules. All scenarios include system-specific task execution sequence, event triggering, message transmission, and so on. The running process of a system can be seen as the change of the scene, and can also be seen as the interaction process within the system. In the IMCL modeling process, the scene layer corresponds to events in the system.
- **Function Layer** The functional layer describes the system’s behavioral process. Relative to the scene layer. The functional layer can be seen as the refinement of the scene layer. It describes the details of the implementation of the scene, including requests for various types of messages that occur in the scene, data calculations and interactions, and the scheduling relationships between the controller and the device.
- **Configuration Layer** The configuration layer reflects the mapping relationship between the computing unit with the control computing capability and the system physical resources in the system. Each computing unit can only control and schedule specific physical resources. The actual description of this constraint relationship is the respective functions of different processing units. By configuring this type of resource constraint relationship, the internal structure of a complex system can be more realistically reflected.

The entire system modeling process mainly includes the following three aspects:

1) **Unified definition of resources** Representations of physical resources are different based on diverse industrial environment, for instance, sensors, read-write devices, and the other resources. Considering their effects on the whole system, we describe all those resources as variables to unifying definition of resources.
2) **Modelling the system** Observing its behavior, the nature of the system is gathering functions, read-write operations, and other actions together. Similar to physical resources, we model them as execution expressions. Multiple execution expressions in one specific order can make up one trigger event
3) **Resource Constraint** It describes the constraints that physical resources are limited available for specific controllers.

B. IMCL formal model

In order to better understand the characteristics of the IMCL model, we introduce the characteristics of the model from a formal point of view. For any system, modeling from the perspective of system trigger events using IMCL language, the system model can be expressed as follows:

$$Prog =$$

From the model refinement point of view, each system $Prog$ program can be seen as a set of trigger events, and then each trigger event T_i is a set of ordered command expressions, each command expression can be seen as a system execution The smallest unit of calculation. For a system, the IMCL language is used for modeling from the perspective of system trigger events. The system model can be expressed as follows:

Definition 1. ***IMCL Model*** $IMCL = \langle V,\ T^*,\ R^*,\ C^* \rangle$

The IMCL model is characterized by event triggering and seen as a combination of events. The system variable V is a set of different variables; T^* is the event set, all events are distributed concurrently; R^* represents the resource definition of the program, which is regarded as a special variable in the model; C^* represents the resources constraint in the system relationship.

Definition 2. ***Variable*** $V = V_{in} \cup V_{out} \cup V_{mess} \cup V_{local} \cup V_{res}$

The set of variables represents how the variable data is expressed in the system. V can be divided into five kinds of sets: the input variable set is represented as V_{in}, the output variable is V_{out}, the local signal variable is V_{mess}, the local variable set is V_{local}, and the physical resource description set V_{res}. These variables run through all state transitions during program execution and involve various types of operational rules for the system.

Definition 3. ***Trigger Event*** $T = \langle id,\ c,\ E^*,\ V_t,\ R_t \rangle$

id represents the unique identifier of the event, each event can represent T_{id}; c is the event trigger condition and it can be a specific conditional expression or a bool value. It is a prerequisite for event execution migration. E represents the set of tasks contained in the event; $V_t = \{Vglobal; Vlocalg\}$ represents the set of variables contained in this event, where V_{global} is the program global variable and V_{local} is the set of local variables inside the event T. $R_t \subseteq R^*$ indicates the physical resources that event mapped. The conditional triggering relationship of an event can be represented by the following expression:

$$(1)\langle id, c, E^*, V_t, R_t \rangle \rightarrow false \Rightarrow \langle id, c^{'}, E^*, V_t^{'}, R_t \rangle$$
$$(2)\langle id, c, E^*, V_t, R_t \rangle \rightarrow true \Rightarrow \langle id, true, E^{*'}, V_t^{'}, R_t \rangle$$

Definition 4. ***Task*** $E = E_{com} | E_{inv} | E_{branch} | E_{loop} | E_{ch} | E_{sh}$

The task represents the unit of program execution. Where E_{com} represents a general assignment operation or numerical operation. E_{inv} indicates that the task is to interact with system physical resources; E_{branch} indicates conditional branch; E_{loop} indicates program loop execution; E_{ch} indicates execution statement of system communication process.

Definition 5. ***Branch*** $E_{branch} = \langle b,\ E_{if},\ E_{else} \rangle$

The branch is the conditional statement, which means that the program enters the selected state. We define b as the branch condition. E_{if} will be chosen if b is true, otherwise the E_{else}. We define $e_0 \in E_{if}$ and $e_1 \in E_{else}$, $\triangle$ as the program execution current state, and $\triangle^{'}$ as the program change state. The program's transition relationship can be expressed in the following form:

$$(1)\ \langle b, \triangle \rangle \rightarrow true, \langle e_0, \triangle \rangle \rightarrow \triangle^{'}$$
$$\Rightarrow \langle if\ b\ then\ e_0\ else\ e_1, \triangle^{'} \rangle \rightarrow \langle e_0, \triangle^{'} \rangle$$
$$(2)\ \langle b, \triangle \rangle \rightarrow false, \langle e_1, \triangle \rangle \rightarrow \triangle^{'}$$
$$\Rightarrow \langle if\ b\ then\ e_0\ else\ e_1, \triangle^{'} \rangle \rightarrow \langle e_1, \triangle^{'} \rangle$$

Definition 6. ***Loop*** $E_{loop} = \langle cond, E_{while} \rangle$

Since the loop statement has been used to indicate that the current program enters a loop execution state, we define b as a loop condition, $\triangle$ as the program execution current state, and $\triangle^{'}$ as the program change state. The execution of the program is as follows:

$$(1)\ \langle b, \triangle \rangle \rightarrow false \Rightarrow \langle while\ b\ do\ e, \rangle \rightarrow \triangle$$
$$(2)\ \langle b, \triangle \rangle \rightarrow true, \langle c, \triangle \rangle \rightarrow \triangle^{''}, \langle while\ b\ do\ c, \triangle^{''} \rangle \rightarrow \triangle^{'}$$
$$\Rightarrow \langle while\ b\ do\ c, \triangle \rangle \rightarrow \triangle^{'}$$

Definition 7. ***Communication*** $E_{ch} = ch!V_{mess} | ch?V_{mess}$

Where $ch!V_{mess}$ that send messages, is the active process of the program. $ch?Vmess$ that the system receives the message, is a similar to the ready-passive process. $\triangle$ is the current state of the program execution, $\triangle^{'}$ is the state of the program change. We represent the transmission mechanism of the communication.

$ch!Vmess$ defines the process as when the model is actively sending out messages, it will change the system message variables, modify the message variables, and the system state will change. The expression is as follows:

$$\langle ch!, V_{mess}, \triangle \rangle \rightarrow \langle ch!, V_{mess}^{'}, \triangle^{'} \rangle$$

$ch?V_{mess}$? defines the process as the model is in the process of accepting the message, and its program state remains unchanged until it receives the target message from the channel.

$$(1)\ \langle ch!, b, V_{mess}, \triangle \rangle \rightarrow false \Rightarrow \langle ch!, b, V_{mess}^{'}, \triangle \rangle \rightarrow \triangle$$
$$(2)\ \langle ch?, b, V_{mess}, \triangle \rangle \rightarrow true, \langle c, \triangle \rangle \rightarrow \triangle^{''}, \langle ch?, b, V_{mess}, \triangle^{''} \rangle$$
$$\rightarrow \triangle^{'} \Rightarrow \langle ch?, b, V_{mess}, \triangle \rangle \rightarrow \triangle^{'}$$

Definition 8. ***Schedule*** $E_{sh} = \langle a_{data}, \lambda, Dev \rangle$

The resource scheduling E_sh reflects the scheduling relationship between the controller and the physical resources. There are two types of λ, $a_{data} \ll Dev$ and $a_{data} \gg Dev$. $a_{data} \ll Dev$ indicates that the controller schedules acquisition of data to the physical device; $a_{data} \gg Dev$ indicates that the controller schedules transmission of data to the physical device. Since the purpose of the IMCL model is to study the logical function of the system, we use the two forms of to describe the scheduling function between the controller and the physical device.

C. Group model generation

The specific implementation details of the population model generation technique have been proposed in our previous published papers. The advantage of using the ICML modeling method is that we can intelligently split a complex model into multiple sub-models given the constraints of the resource and controller constraints. The sub-models can communicate with each other and achieve the same function as the original model. The generated sub-models correspond to specific target platforms respectively, and the main research work in this paper is to realize the code generation work from the IMCL model to different target platforms.

III. Approach

In the previous section we introduced IMCL's model approach to complex systems and the formal definition of the model. Based on this, we will introduce how to generate object code from the population IMCL model.

TODO: fig The approach

The specific implementation details of the population model generation technique have been proposed in our previous published papers. The advantage of using the ICML modeling method is that we can intelligently split a complex model into multiple sub-models given the constraints of the resource and controller constraints. The sub-models can communicate with each other and achieve the same function as the original model. The generated sub-models correspond to specific target platforms respectively, and the main research work in this paper is to realize the code generation work from the IMCL model to different target platforms.

A. Conversion of IMCL Model and Heterogeneous Platform

We conduct research on different platforms, including FPGA, PLC, and PC. The research mainly includes how to use IMCL to represent these heterogeneous systems.

1) ***Conversion of FPGA and IMCL Model:*** FPGA (Field-Programmable Gate Array, Field Programmable Gate Array), due to its customizable features, is widely used in medical equipment, rail traffic control and other fields. The system developed by the VHDL language used by the FPGA includes the following basic parts:

- **Library** Declares the repository that the program needs to use, including the std, work, and user-defined libraries. Library contains a variety of design elements, from a program perspective, can be seen as a collection of data.
- **Use** This section is related to the Library and declares the specific resources used by the corresponding resource library in the Library.
- **Entity** This area is the entity declaration of the VHDL program and mainly describes the relationship between the input, output, and ports of the system circuit.
- **Architecture** The architecture is the behavior part of the circuit. The architecture supports the parallel and serial of the program. The main description is its internal implementation process, including data flow, structure description, behavior description and so on.
- **Configuration** The main purpose of the configuration is to select the required units from the library and form the required system. From the perspective of system resources, it is a process of choice and combination.

TODO fig FPGA-IMCL

Combined with the characteristics of the previously analyzed IMCL model, we can see that there are commonalities between the two architectures. After ignoring the irrelevant platform details, IMCL can model the behavior described by VHDL. As shown in the figure, the library and package structure represented in VHDL can be abstracted into resources in the process of modeling with IMCL after ignoring platform dependencies. The entity represents the design of the circuit structure of the system and can also be used as a description of resources. The architecture in VHDL mainly includes the structure description, data flow description, and system line description. Essentially, they describe the functional characteristics of the internal structure and correspond to the events described in IMCL.

2) ***Conversion of PLC and IMCL Model:*** PLC (Programmable Logic Controller) is a kind of programmable control industrial control computer. PLC takes the microprocessor as the core and realizes the control of the system through software. There are many types of PLCs, but their structural principles are basically the same: they include processors, storage, I/O ports, and network communications. Take the language IEC 61131-3 language by PLC as an example, the design language includes five forms: LD(Ladder Diagram),IL(Instruction List),FBD(Function Block Diagram),SFC(Sequential function chart),ST(Structured Text).

TODO: fig PLC-IMCL

We can use IMCL to represent the operating mode of the PLC. The working phase of the PLC is to periodically scan cyclically, and it will continue to work when there is no interruption or other situations. The general PLC operating mode can be divided into five phases:

- **Internal processing stage**: detecting the system's current ready state and resetting the internal timer;
- **Communication service phase**: The PLC has a communication function. The external control module can communicate with each other and can also receive signal commands from other controllers.
- **Input processing stage**: read the information data of the mounted peripherals and sample the data into the system at one time.
- **Program processing stage**: This stage is the core stage of the PLC control process and is the main body of the PLC program, including condition control, numerical calculation, and logic conversion. This stage reflects the functional behavior of the system.
- **Output stage**: After the main program runs, data is loaded to the outside through the output mechanism.

IV. Rules

The Rules goes here.

V. Case Study

The Case Study goes here.

VI. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.