Template Design and Automatic Generation of Controllers for Industrial Robots

Lenko Grigorov
School of Computing
Queen's University
Kingston, ON, Canada
lenko.grigorov@banica.org

José E. R. Cury
DAS
Federal University of Sta. Catarina
Florianópolis, SC, Brazil
cury@das.ufsc.br

Karen Rudie
Dept. of Electrical and Computer Engineering
Queen's University
Kingston, ON, Canada
karen.rudie@queensu.ca

Steffi Klinge
Institute of Automation Technology
Otto-von-Guericke University
Magdeburg, Germany
mail@steffi-klinge.de

ABSTRACT

The basic theory of supervisory control of discrete-event systems is extended with the notion of *templates*, which simplifies the modeling of controllers since one can work with conceptual designs. In this work, software which provides support for the new design approach is presented along with its application to a robotic testbed.

Categories and Subject Descriptors: J.6 [Computer-Aided Engineering]; J.7 [Computers in Other Systems]: Industrial control

General Terms: Design, Experimentation.

Keywords: System Control, Discrete-Event Systems, Template Design, Code Generation.

1. INTRODUCTION

In an industrial setting with many robots, such as a factory floor, one has to consider higher-level control specifications for the general process of operation, such as robot coordination. The theory of supervisory control of Discrete-Event Systems (DESs) was developed by Ramadge and Wonham [4] in order to address exactly this aspect of control. In this framework, the system of interest is modeled as a finite-state automaton (FSA) which describes its discrete behavior in terms of occurrences of events. The events are instantaneous, spontaneous, and certain control can be exercised by preemptively preventing their occurrence. The entity exercising the control is called a *supervisor*. There is an algorithm that automatically computes the most permissive supervisor for a given specification.

The use of predefined DES units by engineers may lead to a much easier application of supervisory control. In [2], the authors describe how templates are used to design controllers for industrial application. However, the approach is not cast within the supervisory control framework and cannot take advantage of the algorithms therein. This issue is addressed in [3] where templates are defined in the context of supervisory control. There are two types of entities that can be used in a model: modules and channels. Modules are the active parts of the system (e.g., individual robots). Channels are passive parts of the system, specifications that govern the interaction between modules (e.g., network protocols). Common types of system modules or specifications are kept in an abstract form, as *templates*, in a *template library*. Templates are concretized, or *instantiated*, during the process of modeling. If there are many similar robots on the factory floor, the template for this type of robots can be instantiated many times. The designer *links* instances by selecting which events between instances will be considered equivalent.

Here we describe a pilot application of the methodology in practice, i.e., for the control of a small robotic testbed. We developed a prototype tool for template design of DESs and for the automatic generation of Programmable Logic Controller (PLC) code. A more detailed report on this project appears in [3].

2. SOFTWARE AND APPLICATION

From the description of the template design approach, it can easily be seen that the approach borrows ideas from object-oriented programming (OOP). The concepts of templates and instances correspond loosely to the concepts of classes and objects. The links between instances establish how parts of the system interact. However, one significant difference is that the protocols for message passing between objects in OOP have to be determined by control structures in the code. In template design, on the other hand, the protocols for interaction between modules are determined by the behavior encoded in the channels that interconnect them. In other words, the specifications for the whole system are given implicitly. It becomes very easy to substitute one channel with another just by reconnecting the corresponding

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

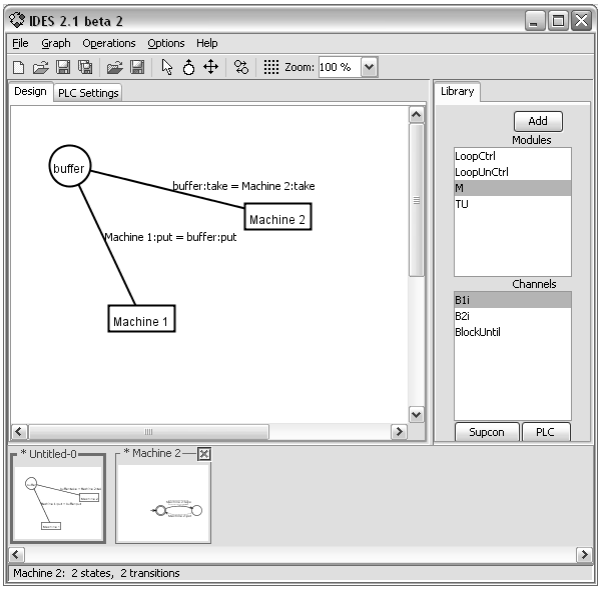

Figure 1: The template design interface in IDES.

modules. That, plus the availability of pre-designed templates in the library, makes the methodology suitable for rapid prototyping of control solutions.

We decided to extend existing DES software to add the template design functionality. The IDES software developed at Rudie's research laboratory, [5], was chosen since its architecture supports the addition of extensions and it offers an advanced graphical interface infrastructure. The latter was an important factor because the user interface we designed is based on the use of the mouse cursor to create and manipulate graphical elements. A snapshot of a template design opened in IDES can be seen in Fig. 1.

The solution of a supervisory control problem is in the form of a supervisor modeled as an FSA. Usually, the events in such a system are abstractions of sequences of low-level events. For example, the event "move robot arm to location 2" may involve a sequence of electric signals sent to one of the motors in order to rotate the required amount. There is a gap that needs to be filled between the abstract controller represented by the DES supervisor and the low-level controller for the real system.

We decided to use a hybrid approach, where most of the control logic is derived from the FSAs of the supervisors, while the user specifies the low-level control code manually. In this way, the majority of the work on developing control software is automated, while the experts retain full access to the hardware. We used the method proposed in [1] to convert FSAs into code. A section of the generated code consists of PLC subroutines to interact directly with the hardware. The execution of a subroutine signifies the occurrence of an event in the control code. This method relies on manual modifications to the generated code in order to insert the required subroutines. Instead of modifying the generated code *post factum*, we propose a simple solution where, for each event in the template design, the user can specify a snippet of PLC code. In other words, the low-level subroutine for each event can be specified within the design. Then, during PLC code generation, this code snippet can be automatically inserted as needed.

In order to test the applicability of template design of DESs and to evaluate the software implementation, the IDES extension was used to design a controller for a robotic testbed at the Department of Automation and Systems, Federal University of Santa Catarina, Brazil. The intended functionality of the system is to retrieve parts from an input buffer, perform operations on the parts and test if the operations were successful. Depending on the outcome of the test, the given part is output into one of a number of buffers (such as "accepted", "reprocess", etc.) The system is controlled via a Siemens S7-200 series PLC unit.

3. DISCUSSION AND CONCLUSIONS

In summary, the template design methodology proved to be much more effective in the design of controllers for our robotic testbed than the traditional method. Overall, the advantages expected from an approach similar to that of OOP were not as important as as the advantages of having a visual environment for rapid prototyping of control solutions. In our project, we needed to obtain a very small supervisory solution (in order to fit in the PLC memory) and it was necessary to go through a large number of iterations where the system was simplified with different approaches. This would not have been feasible if the system had to be completely remodelled every time and if the consistency of the model had to be maintained manually. The automatic generation of PLC code sped up greatly the deployment of each control solution and enabled a faster test cycle.

Future work will focus on modifications to the software so that rapid prototyping is supported better and more freedom is allowed in the customization of template instances.

Acknowledgments

We would like to thank the following people for their help and support: Max de Queiroz, Francisco da Silva, Guilherme Lise and Luis Marques from UFSC, Brazil. The project was supported through grants from NSERC and Queen's University, Canada, and CNPq, Brazil.

4. REFERENCES

[1] M. H. de Queiroz and J. E. R. Cury. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02)*, pages 377–382, Zaragoza, Spain, October 2002.

[2] G. Ekberg and B. H. Krogh. Programming discrete control systems using state machine templates. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 194–200, Ann Arbor, MI, USA, July 2006.

[3] L. Grigorov. Template design of discrete-event systems. Technical report 2007-538, School of Computing, Queen's University, Canada, 2007.

[4] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[5] K. Rudie. The integrated discrete-event systems tool. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 394–395, Ann Arbor, MI, USA, July 2006.