2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.41

Formal Description of a Real-time Component

Qi Zhong-xia*

School of Electromechanics, Northwestern Polytechnical University, Xi'an,710072, China

Abstract. Real-time component model must be effective in its real-time properties specification. It provides mechanisms to ensure component be reused through its interface. On the system design stage, to help developers describing real-time system and its performance, the appropriate description of component property are stressed. Since the formal specification language marked abstract and form characteristic, using formal description descript a real-time component model is suitable.

Keywords: REAL - TIME SYSTEM , FORMAL DESCRIPTION , SOFTWARE COMPONENT

1. Introduction

Real-time software component-based development practice is an important part of the current CBSD (component-based software development). And suitable for use in academia and industry of the existing common component technology in real - time system is PECT[1], Rubus[2], PBO[3], PECOS[4] projects and CORBA[6]. For CBSD research on real-time applications, such as ZHU discussion on using component technology to construction - oriented issues of application specific embedded real-time operating system ASOS[7]; Sheikh implement a component for mobile embedded system of embedded database[8].

Analysis of the existing work of the present study can be found a big disadvantage: research and application is independent of each other. For example, a real-time application software development practices have very little use to the CBSD methodology, development environment, component - based model of research results. The reason lies in the study of CBSD technology of real - time software start and related research is not mature.

In later years, the style of formal approach descripting real-time component specification is various, such as paper [9] define a formal semantics of a component language for embedded systems. And the behavior of a component be described by a timed automaton. Reo[10] is a coordination model for component composition. It defines a very flexible semantics of connectors, where component instances and connection endpoints can migrate during run-time. The paper [11] explains how a declarative method language, based upon the formal notations of Z and B, can be used as a basis for automatic code generation. The paper [12] define a semantics for a new specification language in terms of timed automata, which thus also opens the possibility of analyzing interface descriptions with the model checker. It furthermore give timed simulation conditions and prove their soundness with respect to inclusion of timed traces. Moreover paper [13] provides two contributions that help improve the development, validation, and integration of distributed real-time and embedded systems throughout their lifecycles.

In a formal approach to component specification, interfaces are usually described using pre and post conditions of methods or protocols. In this paper we present an approach for integrating time into a component specification language which already allows for pre/post and protocol descriptions. The

* Corresponding author
E-mail address: qizhongxia@yahoo.cn

specification of timing aspects is indispensable when treating components of embedded systems underlying hard real-time requirements. In order to allow for a smooth integration into the existing specification language and to ease reading and writing of interfaces, we do not extend the language with formalism for time; only add a specific time feature to it. The π calculus is used in description of component behavior. This calculus relation can be used as a correctness criterion for interoperability of components.

Therefore, this article suggests that a simple real-time component model research on formal description. This describes is easy understanding to professional in NC system, that have some help to CNC users development of NC domain system software.

The paper is organized as follows. In the next section, we present our formal model stage by stage. After that, we instance an NC active component for real-time component. The last section is the conclusion of our paper.

2. Model of overall structure

Graphic representation is intuitive and looks straightforward, be very good description of component static connection between the port. But the graphical representation method cannot express service calls between components port of dynamic behavior and behavior constraint relations between the port. So it will need the following text representation. Formal semantics of component model by the BNF pattern syntax. BNF model for component models :

Table 1. Definition of the component

<COMPONENT >::=COMPONENT < Component head > [<Constant >< Variable > < internal collection > < Port > <Component behavior>] END

Component defined by the COMPONENT to start and END to end. There must be a component head, the others are optional part.

Table 2. definition of the component head

<Component head >::= <Component name > [IMPORRTING < Component parameter list > [CONSTRRINTS < Component parameter constraints >]

Component head is a logo of the component. It pointed out that the name of the component. Some cases, apart from the members, Component may also contain component parameter list, behind IMPORTING clause.

The role of component parameter list played that the component parameter list is for members to leave some specified variable, the variable assignment at a time component instance. System provides a component instance to operate NEW. Component parameter lists can be a simple scalar or non - empty collection. Adopting component parameter constraints can be said that the component parameter to introduce some explicit constraint condition. Component parameter constraints is able to accurately explain the scalar formal parameter type, and additional restrictions for separate collection of form parameter

As a basic instance, give cutter compensation component in CNC system for example. Now we know nothing about it.

If the tool has two parameters length and radius， their value reading from the input or an external parameter configuration file or a database in tool compensation starting time. then this extreme simplification model with parameter can be like this formal description :

COMPONENT
cutter compensation
IMPORTING

radius， *length*

CONSTRAINTS

radius∈RAT∧ *radius*>0

length ∈*RAT*∧ *length*>*0*

END

3. Basic Data section

Table 3. definition of constant section

< Constant > ::= CONSTANT < Constant list > [CON-PROPERTY < Constant Property >]

Component constant is characteristic of some of the components remained unchanged, after component instance, this data is not changed. All the constants are simply behind CONSATNTS. You can explicitly specify a constant value, or merely lists a constant and ultimate value in constant component instance, specify are also using NEW operation to achieve an instance. Constants explicitly limiting condition known as constant property.

In tool offset this example, suppose there is a constant R_{min}, said the minimum radius of curvature radius of the parts inside profile. It needs specified at the time of component instance. Cutting Tool radius should be less than R_{min} and take radius = 0.8~0.9 R_{min}, otherwise it would cause, damaged tools and artifacts such phenomena as scrap. Another important constants WCET, said the worst case execution time of component. Now assuming *WCET*=3 μs . In this way, constant partial definitions can be described as :

CONSTANTS

WCET, *Rmin*

CONPROPERTY

radius ∈RAT *radius*≤0.9 R_{min}

WCET=3 μs

Table 4. definition of variable

<Variable >::= VARIABLES < Variable Lists > [INVARIANT < Invariants >] INITIANLIZATION < Operational >]

Variables are part of the internal state of the component, variable value change with formation component internal state changes. First, the variable has a name, it is an identifier that it refers to changes in the components. Secondly, sometimes restrictions introduce to variables explicitly conditions. It is a verb expression under the INVARIANT behind. Finally, each variable must be initialized, the initializes variable column behind INITIALIZATION.

Tool offset above example, assume that a private variable ‘time’ of tool offset, used to record the current time in the process of tool offset. And ‘time’ is always less than the invariant WECT. Examples of variable section can be described as :

VARIABLES

time

INVARIANT

time⩽*WCET*

INITIANLIZATION

time =0

Table 5. definition of internal collection

< Internal collection > ::=SETS < Collection List > [SET-PROPERTY < Properties of Collection >]

Internal collection of the component can be introduced by name, enumeration, or simply a collection, at a time component instance is specified. The collection define clause in a column behind SETS. A properties of collection limited behind SET-PROPERTY, indicating the collection should meet the conditions. Let me cite an example, the tool offset in above example, suppose you have a cutting tool diameter collection DIAMETER that contains the national or international standards standard tool diameter series. It then examples of tool offset in the form of internal collection can be described as :

SETS

DIAMETER=0.1,0.2…

4. method

Table 6. definition of method

< Method>	::= METHODS < Methods List >
< Methods List >	::= < Method Declaration >
< Method Declaration >	::= < Method Head> BEGIN [PRE <Precondition >] <Operational Semantics > [POST<Post Condition >] END
< Method Head >	::= < Method Name> [IMPORTING < Input Parameter List >] [EXPORTING < Return Parameter List >]
<Operational Semantics>	::=< Action Code >

Describes internal dynamic behavior of the components, method is a service provided by the component, the service may be provided to the outside, and it may also be only internal service of the components. Method plays the role is to change the internal state of the component.

Method head is a function identifier that specifies the method name. Sometimes the methods need to be imported in order to perform the appropriate action, these inputs using the input parameter list. Apart from the input parameters, method to provide the results of the operation, which with the return parameter list, ranking behind EXPORTING clause.

In order to ensure that the methods performed correctly, methods may explicitly give a precondition that must be the appropriate action. Of course, If component have internal invariants and pre - conditions include unchanged. Pre - conditions listed in the PRE behind.

Operational semantics of method used to describe the function, that is, the execution of the method will change how the component states. In order to ensure the right of the component states, sometimes executive after the completion of the method body must also make some constraints are met. The constraint method for post - conditions behind POST, It is also a predicate expression

The above example, tool radius compensation in three : no compensation G40, left compensating G41, right compensating G42. Tool length compensation in three Orders : compensation G43 , reverse compensation G44 and abolish the compensation G40. For example, the G41, function of the form defined as :

METHODS

G41

PRE

G17 G18 G19
THEN
transform
END

5. port section

Table 7. definition of ports

<PORT>	::=PORTS PROVIDE < Service Provide Port List > REQUEST < Service Request Port List >
< Port List >	::= < Port >{< Port List >}
< Port >	::=< Port Signature > [PRE< Precondition >] [POST <Post Condition >]
< Port Signature >	::= < Port Name > [IMPORTING < Input Parameter List >] [EXPORTING < Return Parameter List >]

The port is building blocks and other components or external environment for interactive media. Component to offer its services through provide port behind PROVIDE and Requested port behind REQUEST.

Component internal method and external service port bindings. To discuss simply, we will direct the method name as a service provider name of the port, the same name binding. In other words, in the course of providing services, port P through ports R request other component service. Next, we continue the example above. Assuming that there are five tool offset provide port services Pl, P2, P3, P4, P5 Function G41,G42,G43, G44 and G40 where the results of the operation provides out port description :

PORTS
PROVIDE
Pl
PRE *G17 G18 G19*
P2
…
REQUEST

6. Component behavior description

Component contains a wide content, from the perspective of component composition, this article only given related behaviors of component between the port service provides and the port service requests. Behavior is that components have shown external behavior when service provided by external request. In other words, component behavior is the behaviors collection of all of the service provided port. For easy understanding component acts as a whole, component behaviors include all of services provided port. Expression of component behaviors is using by Pi calculus and put behind the BEHAVIOR.

Table 8. definition of component behavior

<Component behavior> ::=BEHAVIOR < π Calculus>

PI- calculus is a description and analysis of calculation model of concurrent systems, has strong ability to describe the process of intermittent interaction between component.

For example, the tool offset component can be expressed as :

BEHAVIOR

P=P1
P1= transform.P1

The process expressions, said P1 implementation process of P1 = transform.P1， that components can perform transformation of repeated operations

7. Discussion

In this paper, we have shown how statements in a formal, predicate notation may be automatically translated to produce executable programs. This is possible because of the restricted form of constant, variable, port and component behavior, which ensures that the value of any updated attribute is completely determined. This might seem to reduce the value of the notation as an abstract specification language, but experience suggests that this is not the case. Before calculating the program semantics of a method, we expand its definition to include contextual information: invariant properties described elsewhere in the model, and assumptions regarding the nature of the application being developed.

It is not unreasonable to expect the author of a specification to include sufficient information to determine the values of any attributes deemed important enough to include in the description of the component state: at least, not if the specification is intended to describe an abstract design.

The real-time requirement of a component based system in general is achieved not only by the individual components but also by their interactions. In order to increase the flexibility for the timing specification of a component, we specify the timing of each of its methods as a relation of the time to carry out the method and the resources provided to the component. The implementation of a method may depend on services from other components which may be mutually exclusive with the presence of concurrency. Therefore, to guarantee its real-time services, a component needs an assumption about the real-time behavior of the interaction of components in the system as well as the schedule for services of the system. To capture these kind of assumptions we introduce connector to the specification of the component interface. Then, the component can provide correct service only if this connector is satisfied. In the literature, there are a lot of work on the component composition scheme, but not many of them take into account the timing specifications to our component.

8. Conclusion

Through component-based development practices of specific real - time application software, we can sum up the problem in these concrete applications, and seek appropriate solutions. To make real-time CBSD technology to CNC software a practical stage. There is still a considerable long roads need to go. For example, in real - time software component based model, cannot well meet the diversity characteristics of real - time software. In terms of the development environment and tools, no attention to software development environment and real - time software customization problems that need to be unified, thus not suited to the characteristics of real - time system development. These problems are hindering CBSD technology application in CNC software development practices obstacles.

9. References

[1] K. C. Wallnau. Volume III: A Technology for Predictable Assembly from Certifiable Components, Technical report, Software Engineering Institute, Carnegie Melton University, Pittsburgh, USA, April 2003

[2] Arcticus Systems Home Page. http:/ /www. -arcticus.se.

[3] D. B. Stewart, R. A. Volpe, P. K. Khosla. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. IEEE Transactions on Software Engineering . 1997 , 23(12):759~776

[4] O.Nierstrasz, G.. Arevalo, S. Ducasse etal. A Component Model for Field Deviece In Proceedings of the First International IFIP/ACM Working Conference on Component Deploylnent, Germany， June 2002.

[5] D. C. Schmidt， D. L. Levine， S. Mungee. The Design of the TAO real-time object request broker. Computer Communications， 1999, 21: 294~324

[6] OMG, CORBA Component Model 3.0, June 2002, http:// www. omg/ org/ technology/ documents/ formal/components.htm

[7] ZHU Lixin,WANG Feiyue. Component- based Constructing Application Specific Transportation Systems, Embedded Operating Systems. IEEE12-15 Oct. 2003, 2 :1338~1343

[8] Sheikh I Ahamed and Sanjay Vallecha. Component-based Embedded Mobile Embedded Systems. In Proceedings of the International Information Technology: Coding and Computing (ITCC' 04) 2004:534~538 April

[9] Jan Carlson. SaveCCM: An Analysable Component Model for Real-Time Systems. Electronic Notes in Theoretical Computer Science. 2006 ,(160): 127~140

[10] Arbab, F., Reo: a channel-based coordination model for component composition, Mathematical. Structures in Comp. Sci. 2004, (14): 329~366.

[11] David Faitelson, James Welch, Jim Davies. From Predicates to Programs: The Semantics of a Method Language. Electronic Notes in Theoretical Computer Science 2007, (184) :171~187

[12] Björn Metzler, Heike Wehrheim. Extending a Component Specification Language with Time. Electronic Notes in Theoretical Computer Science. 2007 ,(176): 47~67

[13] Aniruddha Gokhale, Krishnakumar Balasubramanian, Arvind S. Krishna,et al. Model driven middleware: A new paradigm for developing distributed real-time and embedded systems. Science of Computer Programming 2008, 73: 39~58Louie H , Burns M, Lima C. An introduction and user's guide to the IEEE Smart Grid Web Portal. *Innovative Smart Grid Technologies Conf. Europe (ISGT Europe)* 2010; 1–5.