Contract-based Formal Specification of Safety Critical Systems *

Wei Dong  Ji Wang
National Laboratory for Parallel and Distributed Processing, ChangSha, P.R.China
dong.wei@263.net, jiwang@mail.edu.cn

Abstract

The paper proposes an approach to light-weight formal specification for interfaces, connectors, contracts and integration of component systems based on UML 2.0 superstructure. Both of static and dynamic contracts are provided with formal models. Dynamic contract can be verified through finding a legal environment in an optimistical way for the integrated components, whose contracts are depicted with interface protocol state machines. The consistency for components should be studied. These formal models form the foundation for model checking, compositional reasoning, and real-time architecture development of component-based safety critical systems.

1 Introduction

Component-based software development (CBSD) has been widely applied in safety-critical applications, such as medical instruments, communication and aviation systems. These systems must satisfy some critical properties. It is convinced that the components with comprehensive highly dependable characteristics will be the basis of high confidence software engineering, and the methodology for integrating these components should be studied to obtain the high assurance systems. It will require the formal specification of component, interface, and architecture framework. The corresponding analysis and verification techniques around the safety and reliability properties are also essential. There have been some related works, such as ArchWare project[1], SAVEComp[2], Cadena[3], etc.

Our approach uses component model in UML 2.0 superstructure[4] as the basis of describing safety critical systems. To depict the rigorous context and temporal constraints of interface usage, the light-weight formal specification is attached to component integration through defining interface protocol state machine and its semantics model, namely contract automata. The notions of stateful and stateless are introduced into interfaces to distinguish the specialties of operations. Then the various kinds of essential consistencies are studied from static, dynamic and refinement perspectives, in which system environments are defined in an optimistical approach. The method has been investigated in analyzing SAFE-II, a lifesaving system of spaceship.

2 Component Model and Static Contract

A component interface $u = (O, f)$ is composed of an operation set $O$ and a bool function $f$. $o \in O$ is a stateful operation if $f(o) = true$, otherwise a stateless operation. For clarity, $u!$ and $u?$ can be used for distinguishing provided and required interfaces respectively. A component $(I_P, I_R, P, R, G)$ includes the provided and required interface set $I_P$ and $I_R$, a port name set $P$. Relation $R$ maps ports to interfaces, and the component diagram $G$ describes the internal view of the component. An assembly connector $(u!, v?)$ links a provided interface with a required interface. A provided delegation connector $(p, u!)$ or required delegation connector $(v?, p)$ shows how a signal that arrives at a port $p$ is forwarded to the interfaces on the other side of $p$ for handling. Fig.1 presents an example composite component of fault-detecting in SAFE-II, and $(AO2!, EI?)$, $(pA, EO1)$, $(AI1?, p1)$ are the assembly, provided and required delegation connectors respectively.

In a component diagram $G = (V, F, E)$, set $F$ includes at most one element which is the parent of all components in set $V$. $E = (N_A, N_D, N_D)$ consists of three sets of assembly, provided and required delegation connectors. Each component in $V$ may have its own sub-component diagram from which the hierarchical structure is formed.

The connected interfaces should satisfy some static consistencies to ensure correctly mapping, which can be verified via type checking. For an assembly connector $(u!, v?)$, $v.O \subseteq u.O$ specifies that all operations appearing in required interface should be offered by the provided interface, which is the 1st static consistency. For a component diagram $G = (V, F, E)$ in which component $C \in F$ and port

---

*This work is Supported by National Natural Science Foundation of China Grants No. 60303013, 60233020, 60403048, Open Foundation of State Key Laboratory for Software Engineering Grant No. SKLSE03-08, and Program for New Century Excellent Talents in University.
In a system design of top-down style, when a composite component is decomposed into sub-components, the IPSM $M$ of the composite component should also be decomposed into $M_i$, which are IPSMs of sub-components. The composition of $M_i$ should be consistent with $M$. It implies that $\text{Contr}(M) \geq \otimes \text{Contr}(M_i)$ must be satisfied, where $\geq$ is the well-defined refinement relation.

When constructing the system by integrating the existing components in a bottom-up approach, the contract automaton $A$ of the resulted composite component can be obtained by computing the product of contract automata $A_i$ of each existent component. If the IPSM $M$ has been specified for the composite component in design stage, $A$ should be consistent with the contract automaton of $M$, i.e., $\text{Contr}(M) \geq A$, where $A = \otimes A_i$.

Around these formal specifications, model checking of component-based safety critical systems is being studied, especially combined with compositional reasoning to improve the scalability. Timing constraints are unavoidable in safety-critical systems (e.g. SAFE-II). How to introduce time description and related performance interfaces into verification will be further studied.

References


