

Certification of Distributed Component Computing Middleware and Applications

Sudipto Ghosh

Computer Science Department
Colorado State University
Fort Collins, CO, 80523, USA
+1 970 491 4608
ghosh@cs.colostate.edu

Aditya P. Mathur

Department of Computer Sciences
Purdue University
West Lafayette, IN 47906, USA
+1 765 494 7823
apm@cs.purdue.edu

ABSTRACT

We focus on the issues related to the certification of components and applications conforming to the CORBA 3 standard. CORBA 3 is a standard for Distributed Component Computing (DCC) middleware. Similar standards include Enterprise Java Beans. The specifications for these technologies offer a set of services, such as security, transaction and persistence. Certification of DCC middleware and applications poses several challenging issues discussed in this paper.

Keywords

Distributed object computing, distributed components, Enterprise Java Beans, CORBA services, CORBA Component Model, certification, quality of service, software testing.

1 INTRODUCTION

There is a growing trend towards the development of heterogeneous, large and complex distributed applications on top of advanced *Distributed Object Computing* (DOC) middleware, such as Microsoft's DCOM [13], Sun Microsystems Java RMI [7] and Object Management Group's CORBA [16]. Applications using middleware consist of distributed and interconnected objects. The middleware technologies also provide services such as naming, trading, notification, security, persistence and transaction.

DOC has evolved into *Distributed Component Computing* (DCC). Sun Microsystem's Enterprise Java Beans (EJB) [15] and OMG's CORBA Component Model [1] are examples of DCC. DCC improves on DOC in several ways [14]. In DOC, connections between objects are encapsulated into objects and cannot be configured easily by external software artifacts. The use of system services, i.e. non-functional aspects, must be hard coded into objects and mixed with the functional aspects of the application. DCC middleware provides the use of container servers which host downloadable components and manage system services implicitly. Component implementations need contain only application business

logic. With the use of DCC middleware, applications can be composed from packaged, deployable and distributed components.

In this paper, we describe the issues related to the certification of DCC middleware and applications. We identify the open problems and directions for future research.

The remainder of the paper is organized as follows. Section 2 is short introduction to the specifications in CORBA 3. The issues related to certification are enumerated in Section 3. Existing techniques for certification are summarized in Section 4 and research issues and directions are discussed in Section 5. Section 6 summarizes the paper.

2 FEATURES OF CORBA 3

The specifications in CORBA 3 can be divided into three categories [18]:

1. Internet integration
2. Quality of Service control
3. CORBA component architecture

Internet Integration

The specifications for internet integration are related to 1) Firewalls and 2) the Interoperable Name Service. The Firewall specification defines transport-level firewalls, application-level firewalls and a bi-directional GIOP connection useful for callbacks and event notifications.

The Interoperable Name Service provides a way of accessing remote instances even if its object reference is not known. This service defines one URL-format object reference, `iioploc`, that can be typed into a program to reach defined services, such as the Naming Service, at a remote location. A second URL format `iiopname` invokes the remote Naming Service using the name that the user appends to the URL and retrieves the IOR of the named object.

Quality of Service Control

A number of asynchronous and time-dependent invocation modes for CORBA are defined in the new Messaging specification. Both static and dynamic invocations are allowed to use the modes. Policies allow setting of QOS control of invocations by clients and objects.

There are specifications related to Minimum, Fault-tolerant

and Real-time CORBA. The specifications for minimum CORBA are primarily intended for embedded systems. They do not require the Dynamic Invocation Interface or the Interface Repository. Fault tolerance for CORBA is based on entity redundancy and fault management control. Real-time CORBA standardizes resource control to achieve predictable behavior for both hard and statistical realtime environments.

CORBA Components

The CORBA Component Model presents a container environment that packages transactionality, security and persistence, and provides interface and event resolution. The functions are prepackaged and offer a higher level of abstraction to the programmer than the corresponding CORBA services. Containers keep track of event types generated and consumed by components and provide event channels to transport events. The containers also keep track of interfaces provided and required by the components they contain, and connect one to another where they fit. The architecture provides navigational capabilities through the components and interfaces supported by them.

The model provides a framework that eases the development of heterogeneous components, by providing, for instance, integration with EJBs. EJBs can be installed in a CORBA Components container. CORBA components can be written in multiple languages and support multiple interfaces. The model also provides a software distribution format that enables a CORBA component software marketplace.

3 ISSUES IN CERTIFICATION

Implementations that comply with the new CORBA standard need to be certified to ensure compliance with respect to the specifications for Firewalls, Interoperable Name Service, Quality of Service Control and Component Framework.

Applications that are built to the CORBA Component Model also need to be certified for compliance with the standard. Since these applications are composed of CORBA components, the components themselves need to be certified. This brings up several issues that need to be addressed.

Verification of each component, referred to as *local certification*, is a necessary activity to enable reasoning about the behavior of component-based systems [24]. Components need to be certified for quality, functionality and completeness. As a business may have diverse components, quality assurance personnel need to ensure that these components will work and not be a liability to the company. A component may behave correctly for one set of customers and not for another.

Components obtained from the public domain are usually unsupported while those purchased from a vendor are usually supported. Certifying both kinds of components by an independent third party increases the reusability of these components. Components developed in-house may also need to be certified by the developing organization or a separate quality assurance group. The effort of independent testing required

for certification usually involves a significant resource consumption.

Certification can be performed with different goals. The code for a component can be tested using black-box and/or white-box methods to ensure that it meets the application-specific requirements. Functional attributes also need to be validated. Components may need to be tested for interoperability, portability and adherence to standards in the domain of application. Operational attributes such as performance, efficiency, user friendly interfaces also need to be evaluated.

Issues related to the certification of DCC components and applications are listed below.

1. *Identify configuration for certification:*

By definition, components can be tested in a configuration [20]. In the absence of adequate resources, these configurations need to be fewer but representative of the actual deployment configurations. The identification of representative configurations is not an easy task. The problem of finding representative test data for program testing has been addressed in the testing literature. We need to identify methods for finding representative configurations based on the knowledge of the application domains.

2. *Enabling logging capabilities for tracing:*

A strategy to make certification easier would be to deploy components such that faults leave logged traces [20]. This will enable the tracing of failures in production component systems. Logging services are implemented by distributed object management systems, such as Jiro [19], for resource management.

3. *Availability of source code:*

The techniques used for certification vary depending on the availability of source code. More rigorous certification techniques can be used if the source code is available.

4. *Metrics to quantify the certification process:*

During certification, several metrics are needed to describe the quality of the components, the application and the process of certification itself.

5. *Certifying applications:*

While certifying applications, components need to be composed. Integration and system testing of components is required. The testing technique employed needs to be scalable.

6. *Performance analysis:*

Components presenting identical interfaces may be evaluated on the basis of their performance characteristics. Thus, techniques for evaluating the performance

of components will be useful. Facilities to dynamically create a large number of clients of the components can be used for testing for performance. Profiling techniques are required that provide different levels of granularity starting from components and going down to objects and individual methods.

7. *Verification of real-time properties:*

As applications may have asynchronous messaging capabilities, and real-time property guarantees in the specifications, it is necessary to verify these properties. If errors are located and the code changed, the certification of these properties might become difficult because the timing characteristics may have changed. Techniques for replaying tests will have to be employed.

8. *Certification of security properties:*

COTS components need to be certified for security reasons to ensure that they do not behave maliciously. The components may exploit vulnerabilities in the system. Alternatively, the components themselves may have security holes that others can exploit when the components are incorporated into the system.

9. *Certification of process used to produce artifacts:*

Apart from certifying artifacts, an indirect but complementary approach would be to certify the *process* used to produce the artifacts.

10. *Availability of standards*

Certification standards exist for components in other branches of engineering. We need to develop similar standards related to the certification of products and processes. Organizations like the OMG need to develop these standards for CORBA. This also implies that development organizations should use these standards for certification. However, to develop the standards, several empirical studies are needed to show strong causal links between the product or process quality and the guidelines prescribed by the standards.

4 AVAILABLE CERTIFICATION TECHNIQUES

Since certification involves quality assurance (QA), and testing is an important part of QA, we need to look at the testing techniques that have been reported in the literature. Zhu et al. [25] survey several black-box and white-box techniques used for software testing. Mutation testing [4, 5, 8], object-oriented testing [2, 6, 9, 11, 12, 17] and fault injection testing [3, 10, 21, 22, 23] are other techniques that can be used for software testing. Fault injection technique holds particular promise for certifying components.

5 RESEARCH DIRECTIONS

We believe that research is needed in the following areas.

1. Scalable testing methodologies need to be developed to address the issues outlined in Section 3.

2. Appropriate tools to automate the task of certification. Also needed are integrated test management frameworks that can be used to visualize applications under test and allow definition of events to be monitored and states to be controlled. Implementations based on such a framework would provide logging capabilities for tracing.
3. Regression testing techniques need to be developed for DCC components and applications.
4. Fault injection testing will require the identification of appropriate faults and mechanisms for injection.
5. Test adequacy criteria based on suitable coverage metrics need to be formulated for component and application testing.
6. Identification of appropriate configurations for testing by the developer would become easier if there were general guidelines.
7. Exploratory studies need to be done on the impact of the software process on the quality of the artifacts.
8. Emergence of standards for certifying process and product quality need to be defined by independent organizations and adhered to by the development organizations.

6 SUMMARY

Certification of middleware, components, and applications conforming to the CORBA 3 standard presents several challenges that need to be overcome if reliable software is to be deployed. We have identified these challenges and proposed a research agenda to develop techniques and tools to meet them.

REFERENCES

- [1] BEA Systems et al. "CORBA Components: Joint Revised Submission". Technical Report orbos/99-02-05, OMG, Object Management Group, March 1 1999.
- [2] R. V. Binder. *Testing Object-Oriented Systems Models, Patterns, and Tools*. Object Technology Series. Addison Wesley, Reading, Massachusetts, October 1999.
- [3] J. A. Clark and Y. K. Pradhan. Fault Injection: A Method for Validating Computer-System Reliability. *IEEE Computer*, 28(6):47–56, June 1995.
- [4] M. E. Delamaro, J. C. Maldonado, and A. P. Mathur. Integration Testing Using Interface Mutation. In *Proceedings of International Symposium on Software Reliability Engineering (ISSRE '96)*, pages 112–121, April 1996.
- [5] R. A. DeMillo and A. J. Offutt. Constraint-based Automatic Test Data Generation. *IEEE Transactions on Software Engineering*, 17(9):900–910, September 1991.

- [6] R.-K. Doong and P. G. Frankl. The ASTOOT Approach to Testing Object-Oriented Programs. *ACM Transactions on Software Engineering and Methodology*, pages 101–130, April 1994.
- [7] T. B. Downing. *Java RMI: Remote Method Invocation*. Hungry Minds, Inc, USA, February 2 1998.
- [8] S. Ghosh and A. P. Mathur. “Interface Mutation”. In *Proceedings of MUTATION 2000, (Also to appear in a special issue of the Journal of Software, Testing, Verification and Reliability)*, pages 112–123, San Jose, California, October 2000.
- [9] M. J. Harrold, J. D. McGregor, and K. J. Fitzpatrick. “Incremental Testing of Class Structures”. In *Proceedings of 14th International Conference on Software Engineering*, pages 68–80, 1992.
- [10] M. C. Hsueh, T. K. Tsai, and R. K. Iyer. Fault Injection Techniques and Tools. *IEEE Computer*, pages 75–82, April 1997.
- [11] S. Kirani and W. T. Tsai. “Method Sequence Specification and Verification of Classes”. *Journal of Object-Oriented Programming*, pages 28–38, October 1994.
- [12] D. Kung, J. Gao, P. Hsia, Y. Toyoshima, C. Chen, Y.-S. Kim, and Y.-K. Song. “Developing an Object-Oriented Software Testing and Maintenance Environment”. *Communications of the ACM*, pages 75–87, October 1995.
- [13] S. M. Lewandowski. “Frameworks for Component-Based Client/server Computing”. *ACM Computing Surveys*, 30(1):3–27, March 1998.
- [14] R. Marvic, P. Merle, and J. M. Geib. “Towards a Dynamic CORBA Component Platform”. In *Proceedings of International Symposium on Distributed Object Architectures 2000*, pages 305–314, Antwerp, Belgium, September 21-23 2000.
- [15] V. Matena and B. Stearns. *Applying Enterprise JavaBeans — Component-Based Development for the J2EE Platform*. The Java Series Enterprise Edition. Addison-Wesley Publishing Company, USA, December 29 2000.
- [16] OMG — The Object Management Group. *The Common Object Request Broker: Architecture and Specification, Revision 2.0*. OMG, 1995.
- [17] D. E. Perry and G. E. Kaiser. Adequate Testing and Object-Oriented Programming. *Journal of Object-Oriented Programming*, pages 13–19, January/February 1990.
- [18] J. Siegel. What’s Coming in CORBA 3. *URL* <http://www.omg.org/technology/corba/corba3releaseinfo.html/>, 2001.
- [19] Sun Microsystems. Jiro Technology. *URL* <http://www.jiro.com/>, 2001.
- [20] C. Szyperski. *Component Software Beyond Object-Oriented Programming*. Addison-Wesley Publishing Company, USA, 1998.
- [21] J. M. Voas. Certifying Off-the-Shelf Software Components. *IEEE Computer*, 31(6):53–59, June 1998.
- [22] J. M. Voas. Certifying Software for High-Assurance Environments. *IEEE Software*, 16(4):48–54, July/August 1999.
- [23] J. M. Voas, F. Charron, G. McGraw, K. Miller, and M. Friedman. Predicting how Badly Good Software can Behave. *IEEE Software*, pages 73–83, August 1997.
- [24] B. W. Weide and J. E. Hollingsworth. “Scalability of Reuse Technology to Large Systems Required Local Certifiability”. In *Proceedings of the Fifth Annual Workshop in Software Reuse*, pages 133–141, Larry Lattour, Steve Philbrick, and Mark Stevens, editors,, October 1992.
- [25] H. Zhu, P. A. V. Hall, and J. H. R. May. Software Unit Test Coverage and Adequacy. *ACM Computing Surveys*, 29(4):366–427, December 1997.