Development of Distributed Applications Using Service-Oriented-Architecture

Pragati Priyadarshinee, Y. Ramadevi, and Anuradh Srinivas, *Member, IACSIT*

***Abstract*—There are a large number of software development methodologies for developing standalone software, little effort is being made into investigating specialized methodologies that target the development of Distributed Applications (DAs) in the era of Internet and Web-based applications. Rather than focusing on business models, developers usually spend considerable effort in implementing connectivity between software components that comprise these applications. Since a large number of competing technologies exist, these solutions face serious technology-migration and design reuse problems. This paper advocates approaching the design activity from a business rather than technological perspective by defining a service-oriented software architecture that satisfies the functional requirements in a particular domain. The paper also suggests identifying existing or new design patterns to capture common business process functionalities and fulfill the non-functional requirements.**

***Index Terms*—Design patterns, distributed applications, SOA.**

I. Introduction

A. Objective:

Developers usually spend considerable effort in implementing connectivity between software components that comprise these applications rather than focusing on business models.

A computer programme that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. Distributed applications are used to distribute the data between different applications that are employed to accomplish a task in a network.

B. Methodology:

We concentrate on design activity from a business rather than technological perspective by defining a service-oriented software architecture that satisfies the functional requirements in a particular domain.

In a distributed system, it is common that different hardware systems running on different operating systems over different networks. To use the resources in these environments, software developers may choose different programming languages to implement the client/server or group software pieces. In order to make these software pieces to have the capability and flexibility to interact with each other, they should be developed by following a common distributed component architecture which specifies unified interfaces for method invocation, event handling, fault-tolerance and real-time services. [1]

In this approach besides functional requirements, a distributed component implements functions for publishing its interfaces to let other components know how to interact with this component through a Distributed Component Common Interface, for supporting group communication to satisfy fault tolerance requirements through a Component Replication Interface, and for specifying real-time constraints to its services through a Real-time Specification Interface.

Distributed components can be developed by wrapping these interfaces to the legacy software pieces. We can integrate these components into application software using an integration tool at a late stage.

This tool allows the distributed application developers to adjust the components to offer preferable services and then connect the input and output events and invocations to each other through automatically generated component adapters. To satisfy fault-tolerance and real-time requirements, the integration tool also generates glue-code to maintain component groups and to monitor/schedule real-time invocations. Fig. 1 shows our distributed component-based software architecture composed of the following three basic types of distributed components. [2]

C. Approaches of Developing Software:[3]-[9]

1) Domain-Specific Service-Oriented Architecture

Based on the selected domain and its functional and quality requirements; we investigate and define a suitable service-based software architecture (SA) that facilitates interoperability by abstracting away the underlying technology. This SA must integrate key legacy applications and business processes for that domain.

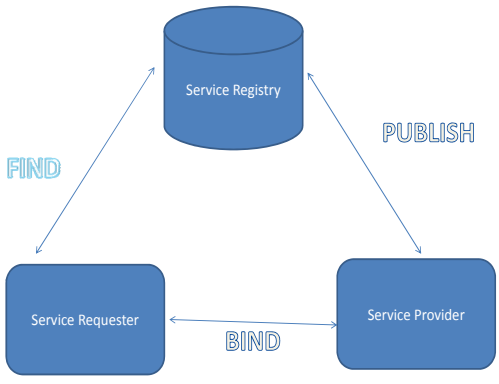

Fig. 1. Domain specified SOA

Manuscript received October 2, 2011; revised December 16, 2011.

Authors are with Chaitanya Bharathi Institute of Technology, Hyderabad (e-mail: pragathi_p@cbit.ac.in; yrdcse.cbit@gmail.com; anuradhabs@yahoo.co.in).

2) Domain-Specific Design Patterns

We address quality (non-functional) requirements through the extraction and identification of candidate design patterns at different abstraction levels. These patterns are solutions to recurring problems that are across different business processes whose SA is derived from 'Step 1'.

Fig. 2. Domain specific design patterns

II. Related Work

1) Qi Yu et al [10] proposed a concrete Web service foundation called Web Service Management System (WSMS) that offers a solid framework for effectively developing, deploying, discovering, publishing, composing, monitoring and optimizing access to Web service. The security/privacy component in the WSMS architecture ensures the implementation of typical security functionalities such as auditing, authentication, access control, and data encryption. While these security issues were described as a summary, none of these issues has been discussed at length.
2) S. Chollet et al [11] presented a tool for supporting and facilitating the integration of Web services. The major security concepts pertaining to Web service composition are authentication, integrity and confidentiality which are handled by login/password authentication.X.S09 certificate and encryption respectively. However, the other important security issues such as authorization of Web services, non-repudiation, etc have not been highlighted.
3) G. Hwang et al [12] proposed an operational model to provide support for securing the Web service transactions. In addition to fulfilling the essential security requirements such as authentication, confidentiality, data integrity and no repudiation, this model provides support for security mechanisms like element-wise encryption and temporal based element-wise digital signatures. This model also supports an explicit key definition which defines three types of keys such as static keys, dynamically selected keys, and keys applied to digital signatures. This model has still limitations and needs to be experimented with distributed enterprise applications.
4) Weiliang Zhao et al [13] proposed trust management architecture for web services to build a trust management layer which extends WS-Trust and covers existing trust mechanisms including credentials, reputation, data storage,etc. Since the primary focus of this work is to provide high level guidelines for the development of trust management solution in web services, the other security issues such as authentication, confidentiality, data integrity and no repudiation have not been exposed.
5) 5)Wei She et al [14] proposed a security model to support the control of information flow through service chains occurred as composite web services. This model also extends the basic security models by introducing the concepts of delegation and pass-on policies to secure interactions in a composed web services. Still this work has provided a foundation for further development of a widespread security model for well-controlled secure web service interactions.

III. Conclusion

The world of software development is witnessing tremendous new developments. It is moving away from traditional software engineering methods towards technology specific methods due to pressures such as fast time to market, stringent quality requirements, the need to integrate legacy applications and the need to support interactions over networks. New approaches such as service-oriented architecture and design patterns provide solutions but their use is still ad-hoc and detached from known software engineering design principles.

This paper investigates an approach that addresses the limitations of existing methodologies especially in the case of distributed applications. First, a service-oriented architecture captures existing legacy systems functionalities in the form of basic services and business processes in the form of composite services. Then a set of design patterns based on this architecture are identified. Their main role is to address both functional and quality requirements. The key idea behind patterns is not to produce new techniques but to provide a framework for the reuse of well-proven solutions that have addressed similar concerns.

References

[1] O. Nierstrasz, S. Gibbs, and D. Tsichritzis,"Component-Oriented Software Development," *Comm. ACM*, Vol.35 No. 9, September 1992,pp.160-164.
[2] S. S. Yau, D. H. Bae, and K. Yeom, "Objectoriented Development of Architecture Transparent Software for Distributed Parallel Systems," *Jour. Computer Communication*, Vol.16 No.5, May 1993, pp. 317-327.
[3] R. M. Adler, "Emerging Standards for Component Software," *IEEE Computer*, Vol.28 No.3, March 1995, pp. 68-77.
[4] J. Montgomery, "Distributing Components," *Byte*, April 1997, pp. 93-98.
[5] R. N. Taylor, N. Medvidovic, and D. L. Dubrow etc, "A Component- and Message-Based Architectural Style for GUI Software," *IEEE Trans. on Software Engineering*, Vol.22 No.6,June 1996, pp.390-396.
[6] Sun Microsystem, JavaBeans Specification,Version 1.0, 1997.
[7] Microsoft Corporation, *The Component ObjectModel Specification*, 1995.
[8] Sun Microsystem, Component-Based Software with Java Beans and ActiveX, whitepaper,http://www.sun.com/javastation/whitepapers/javabeans/javabean_ ch2.html, 1996.
[9] D. Garlan, R. Allen, and J. Ockerbloom, "Why Reuse is So Hard," *IEEE Software*, November1995, pp.17-26.
[10] Q. Y. Xumin Liu, A. Bouguettaya, and B. Medjahed, "Deploying and Managing Web Services: Issues, Solutions, and Directions," *VLDB Journal, Springer,* 2008.

[11] S. Chollet and P. Lalanda, "An Extensible Abstract Service Orchestration Framework," *IEEE International Conference on Web Services (lCWS)*, 2009.
[12] G. H. Hwang, Y. H. Chang and T. K. Chang," An Operational Model and Language Support for Securing Web Services," *IEEE International Conference on Web Services (lCWS),* 2007.
[13] W. Zhao and V. Varadharajan, "Trust Management for Web Services," *IEEE International Conference on Web Services (lCWS),*2008.
[14] W. She, I. L. Yen, B. Thuraisingham, "Enhancing Security Modeling for Web Services using Delegation and Pass-on," *IEEE International Conference on Web Services (ICWS)*, 2008.

Pragati Priyadarshinee received BE from Utkal University in 2004 and M.Tech (SE) degree from Indian Institute of Information Technology, Allahabad(IIITA) in 2008.She is Assistant Professor, Chaitanya Bharathi Institute of Technology, Hyderabad. Her research interests include Software Engineering and Software Architecture. She is a member for IACSIT.

She has two publications one in international Journal and one in international Conference.

Y Rama Devi received BE from Osmania University in 1991 and M.Tech (CSE) degree from JNT University in 1997. She received her Ph.D. degree Central University, Hyderabad in 2009. She is Professor, Chaitanya Bharathi Institute of Technology, Hyderabad. Her research interests include Image Processing, Soft Computing, Data Mining, and Bio-Informatics. She is a member for IEEE, ISTE, IETE, and IE. She has published more than 40 research publications in various National, Inter-national conferences, proceedings and Journals.

B. Anuradha received B.Tech (ECE with Distinction)from S.V.U. College of Engg,Tirupati. and M.Tech (Computer Science with Distinction) from JNTUH. She received her Ph.D. in Computer Science from University of Hyderabad. She is the Principal of Marconi Institute of Technology, Hyderabad since 2008. Her area of interest is Image Processing. She is a life member for ISTE. She has published so many papers in various National, Inter-national conferences, proceedings and Journals. She has 25 years of experiences out of which 5 years in Industry and 20 years in teaching.She received the Best Teacher Award from M.V.S.R. Engg. College, Hyderabad in the years 1993 and 1998.