Sindh Univ. Res. Jour. (Sci.Ser.), Vol: 34 (2) 06 – 10 (2002)

TECHNOLOGY REQUIREMENTS FOR MEHRAN UNIVERSITY VIRTUAL SOFTWARE INCUBATOR (MUVSI) PROJECT

Mushtaque Ahmed Korai, Nasrullah Memon*,Kamal Mustafa Shaikh**,
Zainul Abidin Khuhro***

Department of BS & RS, Quid-e-Awam University of Engg., Sciences and Technology, Nawabshsh

Abstract

The Mehran University Virtual Software Incubator (MUVSI) project provides significant opportunities for teleworking and reduced software development overheads. However few tools exist to address the needs of a MUVSI to support multi-partner distributed software development. Such tools should support Distributed Software Configuration Management (SCM), process modelling and metrics. Certainly there is no unified approach for all these activities. The MUVSI project proposes the implementation of such an integrated prototype MUVSI. In this paper we describe the background to the MUVSI project and discuss some technologies, which may simplify the implementation of such a collaborative environment. The results of this project should make a significant contribution towards the implementation of successful MUVSI environments.

Key Words: MUVSI, teleworking, process modelling.

1. Introduction

A new emerging concept in software production is that of the Virtual Software Incubator (VSI). A VSI is characterized by an often loose and flexible alliance of geographically distributed teams and organizations, with specialization, who come together for a particular, software development [1]. The VSI is one of a new breed of Internet based virtual collaborative environments, which facilitate remote working. As such it provides a new potential market for individual, and groups of, teleworkers.

The emergence of the global Internet has made it possible to remotely access information across organizational and national boundaries. Thus, the concept of a VSI has emerged. MUVSI is in alliance (often temporary) with organizations having different specialisms, which co-operate one another. A number of software development houses benefits from the competitive edge offered by the MUVSI. It provides not only a modern approach to economies of scale but also brings together scarce competencies and resources to meet development requirements [2]. The VSI concept has been made possible through the recent advances in communications technology, which allow the basic infrastructure for communication and co-operation between geographically remote teams of developers. Such environments require software support both in tool provision and in infrastructure support to simplify the implementation of such tools.

With the recent very significant expansion in the use of communication applications, a number of new infrastructure technologies are emerging and evolving. This paper briefly outlines the work of MUVSI project whose core technical objective is to develop a suite of tools to support multi-partner projects geographically distributed across the Internet. The paper then presents a short review of a set of currently emerging communication and of tools to support multi-partner projects geographically distributed across the Internet.

* Department of Computer Systems and Software Engineering, Mehran University of Engg., and Tech. Jamshoro
** Institute of Information Technology, MUET, Jamshoro
*** IMCS, University of Sindh, Jamshoro

CamScanner

The paper then presents a short review of a set of currently emerging communication and infrastructure technologies, which will be considered in terms of their suitability as enabling technologies. It considers these within the context of their potential utilization in the implementation of distributed VSI environments.

2. The MUVSI Project and its Technology Requirements

In very general terms the work in MUVSI falls into four distinct aspects: an initial investigation of the general requirements and technology needed to implement MUVSI (including at this point the development of a Web-based interface to Lifespan); the geographical distribution of multiple SCM (Lifespan) databases and their connection to other SCM tools; the implementation of process modelling in a MUVSI; and the implementation of metrics collection in a VSI. Each stage includes an analysis of relevant user requirements and technology; prototype development and finally user trials.

2.1 Software Configuration Management

Of particular importance in large software development projects is Software Configuration Management (SCM) [3]: the storage, recording and management of all software process deliverable versions and changes. SCM has always been a critical area in ensuring a successful software engineering process and the production of high quality software products. As such it is an integral part of software quality (ISO 9000) and software process improvement standards [4]. As far as using SCM tools within a VSI environment is concerned there are a number of specific requirements, which must be catered for. Examples of the nature of such problems are discussed in technical report [5]: the use of multiple host platforms; the compatibility of tools used at each site; ensuring consistency of replication of module code versions between multiple SCM databases located at different sites; and the need for secure access across networks. It is, therefore, of fundamental importance that the SCM tools used by organizations engaging in VSI collaborations have the ability to address these problems. This means that the tools used will require applying underlying technologies, which can provide solutions to these problems.

2.2 Process Modelling

The process management system in the VSI needs to interface to external process management tools in the same way as module code versions in one SCM database need to be replicated to other SCM databases. It is envisaged that the external process management tools themselves and/or the interfaces between SCM process management and external process management tools will be based on agent technology, which is itself a separate area of investigation [6]. Integration with the tools provided for distributed SCM allows the individual partners in a VSI to collect data and monitor the progress of the software process. This technology will be tightly integrated into the technology supporting the rest of the project.

2.3 Metrics

SCM systems track both changes and their impact on a whole development project. Therefore, the most commonly used software engineering metrics such as the number of defects per module, system or subsystem, can be collected very efficiently by a mechanism with links to a SCM system [2]. However, no SCM systems actually offer metrics collection as a standard feature, although some of those based on commercial relational databases can be adapted to do so. There are no SCM systems, which maintain a selectively

CamScanner

replicated database with true distributed change control and defect tracking and with integrated facilities for metrics collection and analysis.

3. Potential Enabling Technologies for VSIs

The authors have decided, as a central assumption, that Java applets will be used as an enabling technology for its VSI client user interfaces. It follows that the adopted distribution technology will require integration with the Java language (or its extensions) with sufficient industry uptake in order to offer a successful communication mechanism. We have initially assessed several distribution technologies for the implementation of a VSI:

3.1 TCP/IP sockets

TCP/IP [7] is a *de facto* standard connection-oriented transport protocol on which the Internet is built. Sockets provide the programming interface to TCP/IP, albeit a low level interface which does not provide distribution transparency [8]. This lack of transparency means that it is difficult to develop complex distributed applications. Higher-level tools e.g. Java RMI, are now available which are themselves based on TCP (or its connectionless peer - UDP) and offer large productivity gains by avoiding low-level coding detail. However such higher-level tools do not yet have universal acceptance and coding with sockets is still commonplace. It may also be more productive to use sockets to interface to existing legacy applications if only a binary with TCP/IP connectivity is available [9]. Its association with the Internet makes TCP/IP easily identifiable as the single most important communication mechanism for a VSI, but this does not preclude using it to support higher-level protocols.

3.2 HTTP/CGI/Servlets

HTTP [15] is the core protocol for the World Wide Web. It was originally designed for static hypertext document transferring, but has been used increasingly for dynamic Internet applications in which transferred HTML documents are dynamically generated by CGI scripts at WWW servers. CGI [19] is now the established way of extending WWW servers to provide dynamic content, typically for connection to background database servers. However HTTP/CGI has been criticized [10] as being unable to maintain state between client-server invocations, has a poor parameter passing mechanism and is very slow. It is therefore unlikely to be a candidate technology for the next generation of distributed applications. As an alternative to CGI based scripts, Java Servlets [16] are a more recent technology providing a more scaleable and easier way to create dynamic and interactive WWW applications.

3.3 CORBA

OMG's CORBA [17] is an open standard specification for the implementation of object-oriented distributed systems written in many languages and running on many architectures and operating systems. OMG are finalizing a Java binding standard, which allows applets to use services provided by remote CORBA objects [18]. The CORBA standard specifies an interoperable communication framework (usually through its IIOP protocol) and language bindings. The combination of distribution transparency and the application of object-orientation to distributed systems makes CORBA an attractive candidate for the next generation of complex distributed applications, i.e. much of the detailed coding of networked applications can be avoided. The application of an object-oriented paradigm

CamScanner

brings with it attractive features such as encapsulation and reuse. Higher level services, e.g. asynchronous message passing, lifecycle and security although standardized may not be available as implementations in the marketplace. Others may still be in the process of standardization. CORBA's uptake has so far been uneven: the lack of higher level services, cost of development tools, lack of clear advantage in its adoption and Microsoft's marketplace dominance are seen as contributing factors.

3.4 ActiveX/DCOM

Microsoft's ActiveX [11] and DCOM [12] technologies form a set of *de facto* standards. DCOM provides a binary standard for the communication of networked components; the networking technology is based on OSF's DCE RPC (Remote Procedure Call) [13]. Microsoft's philosophy has been to extend its single host COM technology with an existing RPC implementation thus leveraging the large software base that already exists in OLE objects and controls. DCOM is not object-oriented in the traditional sense; it is generally considered more difficult to write applications in DCOM than in CORBA. DCOM implementations are now arriving only on non-Microsoft operating systems. Visual J++ provides a mechanism for writing Java applets that act as clients to DCOM servers. Although Microsoft are intending to introduce better security for DCOM in NT 5.0 they do not as yet consider higher level services such as object migration or multicasting.

3.5 Java RMI

JavaSoft's Java JDK 1.1 language extensions [14] include a RMI (Remote Method Invocation) package. This provides additional functionality allowing Java applets to call methods provided by remote Java objects. This facilitates direct communication between any machines supporting a JVM (Java Virtual Machine). Java provides object serialization across a RMI call to implement object migration. JavaSoft is continuing to develop RMI and is intending to offer advanced services such as security and message passing. A CORBA client-side binding is included in JDK1.2 to allow communication with non-Java servers.

4. Conclusions

MUVSI will provide the support environments for the next generation of software engineering projects based on dynamic, geographically distributed, multi-partner virtual collaborative environments. Only now, with the development of the Internet and sufficiently mature development technologies, it is possible to apply distributed systems to the support of such projects. It is likely to change the human environment within which software development takes places, with a move towards significant use of remote teleworking.

We have undertaken an initial review of technologies supporting distributed applications development, and evaluated how these technologies can be applied to meet the likely requirements of a VSI. Each of these solutions has also to be judged on its merits for specific legacy tools.

5. References

[1]. Belady L.A., From Software Engineering to Knowledge Engineering: The Shape of The Software Industry in The 1990s, International Journal of Software Engineering and Knowledge Engineering, 1(1), pp1-8, 1991.

CamScanner

[2]. Boldyreff, C., Newman, J., Taramaa, J., Managing Process Improvement in Virtual Software Corporations, IEEE 5th International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'96), Stanford University, California, USA, June 19-211996.

[3]. Babich, W.A., Software Configuration Management, Addison-Wesley, 1986.

[4]. Paulk M.C, Curtis B., Chrissis M.B. & Weber C.V., Capability Maturity Model, Version 1.1, IEEE Software, 10(4), pp18-27, 1993.

[5]. Foley, R., Haag, Z., Conclusions of an Assessment of Configuration Management Tools for VISCOUNT, Computer Studies Technical Report COS/CSCW/02/1998 – Glasgow Caledonian University.

[6]. Mair, Q., Agent Technology in a VSI, Computer Studies Technical Report COS/CSCW/03/1998 – Glasgow Caledonian University.

[7]. Halsall, F., Data Communications, Computer Networks and Open Systems 4th ed., Addison Wesley, 1995.

[8]. Coulouris, G., Dollimore, J., Kindberg, T., Distributed Systems - Concepts and Design (2nd. Ed.); Addison-Wesley, 1994.

[9]. Mowbray, T., Zahavi, R., The Essential CORBA; OMG/Wiley, 1994.

[10]. Orfali, R., Harkey, D., Client/Server Programming with Java and CORBA, Wiley, 1997.

[11]. Chappell, D., Linthicum, S., ActiveX Demystified; Byte, September 1997.

[12]. Brown, N., Kindel, C., Distributed Component Object Model Protocol -DCOM/1.0 Draft Specification, 1996.

[13]. Rosenberry, W., Kenney, D., Fisher, G., Understanding DCE, O'Reilly, 1992.

[14]. Horstmann, C. S., Cornell, G., Core Java Volume 2 - Advanced Features, The SunSoft Press, 1998.

[15]. W3C,http://www.w3.org/Protocols/Overview.html, 1997

[16]. Sun Microsystems, http://java.sun.com/features/1997/aug/jws1.html, 1997.

[17]. OMG, CORBA/IIOP 2.1 Specification; OMG document 97-09-01, 1997.

[18]. OMG, IDL to Java Mpping; OMG document 97-08-06, 1997.

[19]. Nasrullah Memon et al, Distributed Information Access for E-Commerce Applications, Sindh University Research Journal 34(1), 9-18, June 2002.

CamScanner