DOI: 10.1017/S1431927603446333
Microsc Microanal 9(Suppl 2), 2003

Using Distributed Software Techniques to Build a Remote Microscopy System

Samuel Santiago and Christopher Morgan*

*Department of Mathematics and Computer Science, California State University, Hayward 94542

As remote control of electron microscopes progresses from novelty to common practice, various approaches to software and hardware delivery systems have been and will be developed. Production systems will need to provide responsiveness, reliability, real time behavior, security, scalability, and cooperation among heterogeneous operating systems and architectures. These requirements are shared by modern commercial distributed systems for delivering commerce over the Internet. The basic infrastructure of the Internet is growing to meet these demands by gradually replacing older dialup networks and moving to broadband technology that stresses quality of service. The acceptance of standard protocols such as HTML, HTTP, and TCP/IP has given rise to an era of high cooperation among systems with different operating systems and architectures. These protocols have been able to minimize the differences between systems and provide a common language for inter-system communication. The Java 2 Enterprise Edition (J2EE) has emerged as a standard for designing and implementing distributed systems mainly for electronic commerce. Supporting its use are software methodologies and technologies such as the Rational Unified Process (RUP), the Unified Modeling Language (UML) and design patterns [1]. This paper discusses how J2EE can successfully be applied to remote microscopy. An actual system called the Java IRSA Gateway (JIG) [3] was developed by one of the authors to control a Philips XL 40 Scanning Electron Microscope. It is in use at the Microscope and Graphics Imaging Center (MAGIC) at California State University, Hayward [2].

J2EE was developed and offered to the larger computer community by Sun Microsystems to provide a framework for distributed systems. A number of software vendors support it, including the free source Jboss. J2EE defines multiple tiers – the Client Tier, the Web Tier, the Application Server Tier, and Data Tier – that can be geographically dispersed (see Figure 1). Each tier can be housed in a collection of containers that support individual components. These components represent external users and other entities such as databases and instruments (such as a microscope), as well as interactions among these entities at various levels of abstraction. This component-based approach allows system behavior to be separated in coherent packages of reusable pieces.

The Unified Modeling Language (UML) helps map human-oriented models of the system into functioning software. UML is a standard supported by the Object Management Group (OMG). Interactions among entities such as instrument users, instrument operators, instruments, specimens, and databases can be represented as diagrams. Various types of diagrams depict behavior such as logging into a session, adjusting an instrument control, capturing an image, recording an action, and logging out. The process of generating these diagrams (and even code) can be facilitated by tools such as Rational Rose and managed by software methodologies such as the RUP.

The J2EE-based architecture has provided several benefits to the MAGIC project at CSU Hayward. Future project developers will have a widely-accepted, well-supported, and standard distributed system architecture as a foundation. J2EE will provide a framework for designing future enhancements, allowing a uniform approach for the design and development of any new components or subsystems. New issues and developments in distributed computing can be incorporated in the microscope control program by taking advantage of J2EE's container-based approach. The J2EE specification is evolving to address distributed computing issues in response to feedback from developers identifies deficiencies or additional requirements. The vendors incorporate these enhancements into the newer versions of their product in order to maintain conformance with the J2EE standard. The JIG can take advantage of these new enhancements as desired to implement new functionality with minimal software architectural redesign.

The JIG has laid the foundation for future evolution of MAGIC/IRSA. Further potential enhancements include integration of a relational database for scheduling use of the SEM online by users, collecting SEM usage statistics, and allowing users to store the images of their samples for historical purposes. Collaborative specimen viewing is another potential enhancement. This concept would allow a single user to control the SEM, but allow the user to give other users permission to view his images simultaneously. Thus a group of geographically dispersed users could collaborate on the examination of images generated by the SEM.

The JIG architecture is a solid illustration of how a modern software concept, design patterns, and a relatively new technology, J2EE, can be applied with a process, the RUP, to solve a real-world problem of designing a remote instrument interface [3]. The JIG has been deployed successfully across multiple environments and configurations in the IRSA laboratory and is currently serving as the active gateway for the MAGIC facility.

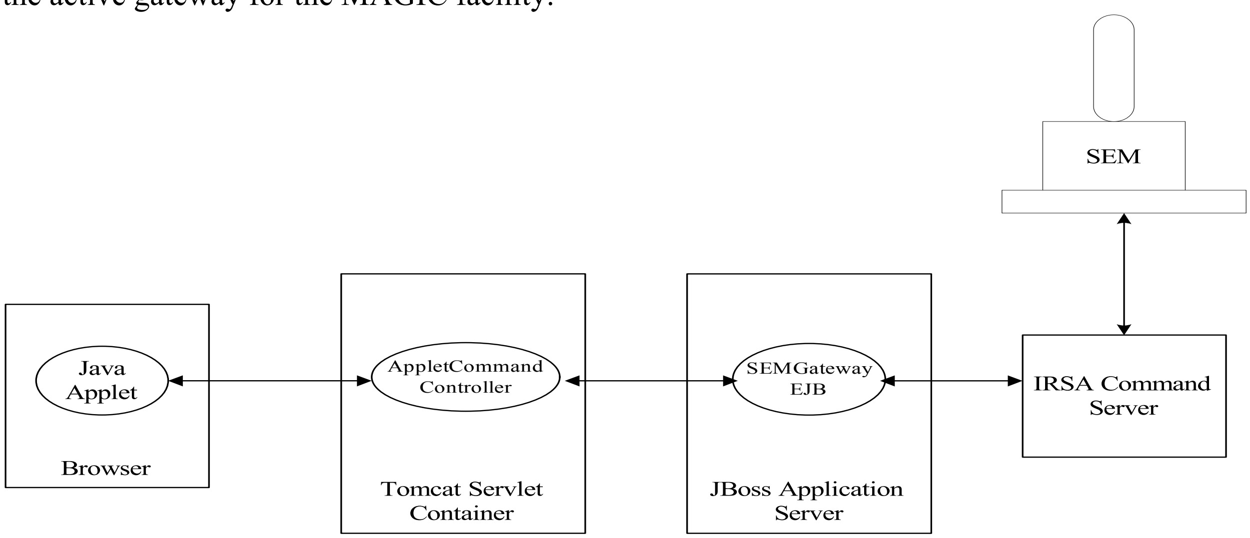

Figure 1: Overview of JIG software architecture

[1] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley, 1995.

[2] C. Morgan, D. Pardoe, and N. Smith, "Toward a Standard for Remote Microscope Control Systems", *SCANNING*, 20:110-116, 1998.

[3] S. Santiago, *An Extensible Software Architecture for the IRSA Gateway*, Masters Thesis, Department of Mathematics and Computer Science, California State University, Hayward, 2002.

https://doi.org/10.1017/S1431927603446333 Published online by Cambridge University Press