Proceedings of Student/Faculty Research Day, CSIS, Pace University, May 4th, 2007

Presenting a JUnit Testing Framework to a Multi-University Community

Romerl Elizes

Research Administration Systems, University of Medicine and Dentistry of New Jersey
1 World's Fair Drive, Somerset, New Jersey 08873 (732) 235-5912
elizesrc@umdnj.edu

Abstract

Coeus is an Electronic Research Administration (eRA) system developed by the Massachusetts Institute of Technology (MIT) to automate the proposal creation and grant tracking processes of participant research universities. Participant universities through an annual nominal fee purchase a subscription to the Coeus Consortium sponsored by MIT that allows them to download the source code and binaries of the application and use them for their research needs. Subscribers are allowed to modify the code and potentially contribute their work back to the Consortium similar to an Open Source community. However the application testing process only involves MIT internal testing of the application and a one week testing process by a select group of stakeholders prior to a major release of the product. This paper not only proposes a testing community framework targeted to the body of universities that are major stakeholders of the Coeus application, but it attempts to define a framework that maybe beneficial to other community-based technology frameworks.

Introduction

The Coeus Application is an Electronic Research Administration application that is developed by MIT [COE]. The Coeus application is a comprehensive application framework that attempts to address all of the needs of the Research Administration function of a university by automating processes related to proposal development, proposal tracking, grant tracking, pre and post-award accounting, internal review boards, conflict of interest, and federal compliance issues. MIT has constantly evolved Coeus to address many concerns in the research administration landscape. One of these major concerns is the National Institute of Health requirement for grant applications to be submitted electronically rather than manually starting with the 2007 fiscal year. While many university research offices have been clamoring to meet this major requirement, they find it difficult to acquire an automated process that can help them expedite all of their research support activities.

Research Administration is a vital function in the university environment. It is the conduit between actual scientific research and the funding required to support this research. The Research Administrators help researchers fill out submissions for their grant proposals and navigate through the various issues related to research: Internal Review Boards, Conflict of Interest, electronic submissions, specific grant requirements, award budgeting, and many other research accounting concerns. Because of the overwhelming responsibilities of this function, Research Administration requires some automotive processes that would help expedite these processes.

MIT produced the Coeus eRA with this automation in mine. The business model behind the deployment of the application is even more comprehensive. While MIT charges a nominal fee for annual subscription to its product for each entity, it allows the entity to download its code and binaries so that the recipient entity can customize and deploy the application according to the entity's needs. The Coeus eRA is divided into several key modules: Proposal Development, Award Budget Tracking, Conflict of Interest, Internal Review Board, System to System submissions, and others. Each recipient university utilizes only a certain number of modules for their implementation of Coeus based on their research faculty's immediate needs. The entire set of universities that are participants of the Coeus business model is called the Coeus

Consortium. MIT invites individual universities to contribute their customization projects if it is beneficial to entire Consortium. The Coeus Consortium follows an open source community model.

One major disadvantage of the MIT implementation of the Coeus eRA is the testing framework behind the application. The Coeus staff utilizes two testing mechanisms: an internal testing mechanism performed by the developers and a one week testing cycle involving select members of the Coeus Consortium. While the testing mechanisms attempt to expose all bugs and upgrade issues in the product, the testing cannot catch all of the issues in the product. Considering all the new software development that the MIT Coeus team is compelled to perform in tight time frames, a more comprehensive testing framework should be developed that attempts to cover all aspects of the Coeus product during development and deployment.

JUnit Testing Framework and Coeus

The author was initially tasked to enhance the Coeus product at the University of Medicine and Dentistry of New Jersey (UMDNJ) by introducing Business Intelligence (BI) reporting capabilities into the product. The Coeus product does not have charting and reporting capabilities and UMDNJ Research Administration felt that this capability would provide meaningful, global and departmental snapshots of proposal and grant activity. Based on work the author did with JUnit methodologies at his previous employment and at Pace University, he implemented a basic JUnit testing infrastructure that easily tested the Coeus application from two aspects: a Cactus framework that specifically addresses database table and web page content validation; and an Abbot framework that uses robot automation to test the validity of GUI components. JUnit, Abbot, and Cactus frameworks are open source tools used specifically for testing of application code.

The initial infrastructure was so successful within the UMDNJ infrastructure, that the author presented his findings to the Coeus User Conference. In one of the presentations, the author demonstrated the capabilities of the JUnit testing framework that was developed at UMDNJ. The author also specified the need for a comprehensive testing framework that could benefit the Coeus application. Understanding that the MIT Coeus staff had limited resources, he suggested a Coeus Consortium testing framework, a community of testers, that would utilize their individual strengths in developing a similar testing methodology that could be shared throughout the Coeus Consortium. UMDNJ has a testing framework for proposal tracking, grant tracking, reporting capabilities, database table validation, and web content. The author proposed that this testing framework could be introduced into the MIT Coeus infrastructure. Other Coeus members who have their strengths within the individual Coeus modules such as proposal development, internal review board, and conflict of interest can develop testing for their modules following the proposed framework. Since the Coeus Consortium encompasses over a hundred participating universities, such a testing framework would encompass a whole variety of scenarios that would engender superior product quality and validity.

Literature Review

Although the paper focuses on a "Community of Testers," (CoT) the idea came from the theory of Communities of Practice (CoP). CoP suggests the concept of team infrastructure and multiple overlapping communities for sharing knowledge and standardizing practices [KAH]. While CoP focuses on Communities within a specific location using workshops and team building to foster collaboration, the concept cannot easily be applied to a community of universities which have distance and timing factors that can affect collaboration. The CoT concept has elements of Fowler's Agile community [FOW]. As individual entities, each university has its own ideas, leaders, and communities. As a Consortium, each university borrows from ideas and techniques from one another.

Some research has been done on Open Source Communities and how they have influenced the software development landscape. Corporations and political organizations have tried to emulate the Open Source model with varying degrees of success and challenges. PyPy is an open source project whose primary customer is the European Union (EU) [DUR]. The Open Source community behind PyPy developed a product with EU funding and attempted to merge agile practices with traditional waterfall development paradigms of the EU. This implementation faced many cultural

and development challenges. In a corporate example, Gurbani of Lucent Technologies attempted to develop an Open Source community for a Telecommunications Signaling Server within the confines of Lucent Technologies [GUR]. Like PyPy, cultural hurdles had to be addressed to make the project successful. In separate research, Crowston offers advice to potential contributors what they need to know about the health of the Open Source community they want to participate in [CRO].

From a quality assurance (QA) standpoint, work has been done to establish a framework for the Open Source Model [KOP] and describing the QA process in corporate environments [MAK]. However, none of these papers indicate successful implementation of these processes in an actual work environment.

Relevance

This work is important for six distinct reasons: 1. the work involves an actual institution with actual stakeholders that will benefit from the introduction of this feature, 2. the work proposes a solution based on limited or no funding that many educational institutions must contend with, 3. this work proposes a traditional development model with some agile influence, 4. the concept of community of testers, 5. the marriage of software development and testing, and 6. support of the technology on research administration. The last three reasons are of significance to this work.

Community of Testers. The work proposes the existence of a community of testers. The literature review did not indicate an actual Open Source community existing solely for testing purposes. This work shows that an actual model can be developed and provide some meaningful relevance to Open Source methodologies and QA practices.

Marriage of Software Development and Testing. Development and testing based on Agile practices should go hand-in-hand. This work furthers the concept of development and testing as part of the same initiative. This work proposes that developers take a proactive role in testing.

Research Administration Information Technology (RAIT). The work furthers the development of RAIT. Based on the literature review, there is no work done to support RAIT. With many other disciplines such as Biology merging with IT, it would some logical that research administration merges with IT as it requires some of the automotive processes that IT has to offer.

Methodology

Initially, the Coeus Application at UMDNJ was deployed with the following configurations:

- Tomcat 5.0.2.8 Application Server – on Linux OS
- Oracle 9G Database Server – on Linux OS
- Oracle 9G Client on Windows, MacOS, and Linux client machines
- JDK 1.4.2.13 – on both server and client machines.
- Coeus Application – Java Struts-based application on Business and Web tiers; Java Swing application on Client tier deployed via Java Web Start.

The Coeus Application was deployed on four Linux servers each with the similar configuration listed above: Production, Backup, Test, and Development Servers.

Testing methodologies. Based on previous work under Dr. Fred Grossman and Dr. Joseph Bergin's tutelage, the author implemented a two-tiered testing infrastructure. Both tiers were based on JUnit testing methodologies.

The first tier testing infrastructure (FTTI) focused on testing the Web and Business Tier logic of the Coeus application. It utilized the Jakarta Cactus framework and focused on database table validation, query output count, web page login validation, and report servlet content. The Jakarta Cactus framework is an open source application framework written in Java that allows for comprehensive Server-side application testing. The FTTI was compiled and deployed on a Tomcat Application Server and was testing against the Production, Backup, Test, and Development Servers. The author developed 50 primary tests from which most of them were database-specific. The Cactus framework was modified to place more useful information on the Cactus test output screen (Figure 1).

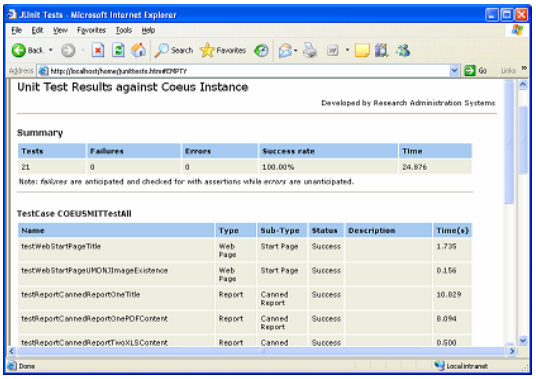

Figure 1. Jakarta Cactus Test Output

The second tier testing infrastructure (STTI) focused on testing the Client Tier logic of the Coeus application. The STTI heavily uses OS console work in MS-DOS and Linux, but its primary application is the Abbot framework. Abbot is a comprehensive application framework that allows for unit testing of Java GUI components. It provides a robust robot testing mechanism. It features a rich script editor called Costello (Figure 2) that allows for the scripting of GUI interactions.

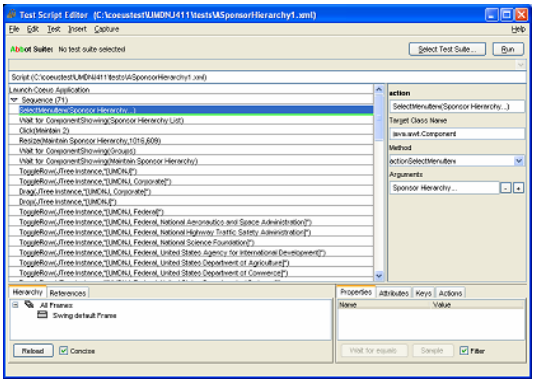

Figure 2. Abbot – Costello GUI Script Editor

The author focused on scripting 20 GUI interactions for the testing environment. The GUI interactions being testing included new reporting functionality for the extended application, testing sponsor hierarchy existence (Figure 3), testing award creation and access, and testing proposal creation and access.

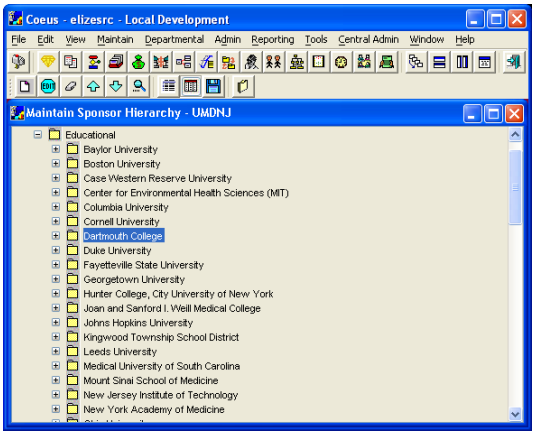

Figure 3: Abbot Testing Sponsor Hierarchy

Discussion

The author of this paper presented the methodology of this work at the Coeus User Conference. In addition to the JUnit framework that the author worked on for UMDNJ, the author specifically asked the Consortium to accept this framework. The framework would be beneficial to the Consortium and it would positively impact the evolution of the product.

The author indicated that the UMDNJ staff is too small to handle a comprehensive testing of the application. As indicated previously, each of the universities has a specific strength in developing and testing certain modules of the product. Consider the following scenario: there are 50 universities in Consortium X and each university had only one developer/tester on staff to test the product. If each developer/tester developed a Cactus/Abbot test that was beneficial to Consortium X once per week for 50 weeks per year, then 2,500 robust tests were made for Consortium X. However, if the developer/tester developed one comprehensive test per work day to arrive at 200 tests for a one year period, that would mean 10,000 tests for Consortium X in one year. The actual Consortium product with its multiple releases, multiple application deployments, multiple modules, and divergent customer implementations would stand to benefit from 10,000 tests.

The author proposed the creation of the Coeus Testing Community which would follow the Open Source model. Participant universities can actively participate in the testing development of the product and include the tests that they feel are important for their realm of expertise.

Many leaders of the Coeus Consortium including several universities expressed a very positive interest in the testing framework and were anxious for its creation in the future. A testing framework shared by the entire Consortium would yield substantial dividends to the development of the product and gain member confidence in the product's integrity.

Future Work

This is an ongoing endeavor. Preliminary consensus and approvals between MIT and the author to establish this testing infrastructure will take some time. While excitement was certainly generated at the Coeus User Conference for this testing framework, until this testing framework actually becomes live, it is only a promise, not a reality. The intended deadline for this implementation is March 2008, however. The framework will be a similar model discussed in the Methodologies section but will encompass multiple releases of the Coeus product rather than multiple database instances. Some security measures and permission issues will be established as this application is developed and deployed. The successful implementation of the framework will be called the Coeus Testing Community.

Upon deployment of the testing framework, the author and relevant stakeholders will be tasked to invite all participant universities to contribute to the Coeus Testing Community. Some of the universities with critical functionality requirements will be active contributors to the Coeus Testing Community. Subsequent research could highlight the results and consequences of implementing the Coeus Testing Community.

While the Cactus and Abbot frameworks are viable testing mechanisms for the Consortium, there is room for other open source methodologies and testing frameworks that would benefit the Consortium. Research on the viability of these alternative tools would not only be beneficial to the Consortium, but to the field of quality assurance as well. Moreover, more research on other community-based testing frameworks would benefit this "community of testers" concept. The current research does not focus on any open source communities whose primary function is quality assurance.

References

[COE] "Coeus Consortium." Web site. Massachusetts Institute of Technology. 2007. Link: http://www.coeus.org/coeus-cons/

[CRO] K. Crowston, J. Howison. "Assessing the Health of Open Source Communities." In *Computer, pp. 89-91*. May 2006.

[DUR] B. During. "Trouble in Paradise: the Open Source Project PyPy, EU-Funding, and Agile Practices." In *Proceedings of AGILE 2006 Conference (AGILE'06), pp. 221-231*. July 2006.

[FOW] M. Fowler. "The New Methodology." Martin Fowler website. December 12, 2005. Link: http://www.martinfolwer.com/articles/newMethodology.html

[GUR] V. Gurbani, J. Herbsleb, A. Garvert. "A Case Study of a Corporate Open Source Development Model." In *Proceedings of the Twenty-Eighth International Conference on Software Engineering (ICSE'06), pp. 472-481*. May 2006.

[KAH] T. Kahkonen. "Agile Methods for Large Organizations – Building Communities of Practice." In *Proceedings of the Agile Development Conference (ADC 2004), pp. 2-11*. June 2004.

[KOP] T. Koponen. "Evaluation Framework for Open Source Software Maintenance." In *Proceedings on Software Engineering Advances (ICSEA'06), pp. 52*. October 2006.

[MAK] P. Maki-Asiala, M. Matinlassi. "Quality Assurance of Open Source Components: Integrator Point of View." In *Proceedings of the Thirtieth Annual International Computer Software and Applications Conference (COMPSAC'06), pp. 189-194*. September 2006.