

TECHNIQUES FOR INTEGRATING PHP AND JAVA APPLICATIONS

BY

CONRAD NOBERT

A project submitted in partial fulfillment

Of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta

December, 2007

© Conrad Nobert, 2007

DEDICATION

To Rechel. My heart is yours.

ABSTRACT

Enterprise application integration is an important activity in today's medium- and large-sized organizations. Organizations have an opportunity to leverage open source content management system software in their integration projects. However, while the best of these content management systems are written in PHP, many organizations have expertise primarily in Java technology. They would benefit by understanding the opportunities and risks involved in integrating their Java applications with PHP. This paper investigates techniques for integrating PHP and Java. The author integrated a web-based subsystem written in Java with Drupal, an open-source, PHP-based content management system. Different subsystem use cases were integrated with Drupal using three different techniques: Interface integration (screen scraping), a socket based integration technique (PHP/Java Bridge), and Web services. The advantages and disadvantages of each technique were analyzed. All three techniques are appropriate in certain circumstances. Organizations should consider integrating PHP and Java applications, and when doing so they should consider the three integration techniques. For the majority of situations, Web services are the most easily leveraged, and therefore the of most appropriate integration technique to choose.

ACKNOWLEDGEMENTS

I would like to acknowledge and thank my project supervisor, Richard S. Huntrods, for guidance and wisdom provided. I would like to also acknowledge and thank my employer, the Northern Alberta Institute of Technology, for investing in me. Most importantly, I thank my family. Jacob and Luc, your smiles and laughter made every day bright. Rechel, your encouragement and support made this undertaking possible.

TABLE OF CONTENTS

CHAPTER 1.....	1
INTRODUCTION	1
<i>Statement Of The Problem</i>	1
Business Opportunity	1
Technical Problem	2
Impact of the Problem/Opportunity.....	3
Potential Causes of the Problem/Opportunity	5
Potential Solutions to the Problem/Opportunity.....	6
CHAPTER 2.....	9
REVIEW OF RELATED LITERATURE	9
<i>Integration Project Types</i>	9
<i>Best Practices</i>	12
REVIEW OF ORGANIZATION DOCUMENTS	15
<i>Ecology Systems Information Society</i>	15
<i>GreenEdmonton.ca Concept</i>	16
<i>GreenEdmonton.ca Technical Considerations</i>	16
CHAPTER III	18
RESEARCH METHODOLOGY	18
<i>Research Methods</i>	18
Metrics-based Approach	18
Action Research and Qualitative Analysis	21
<i>Development and Deployment Environment and Tools</i>	23
Software	23
Hardware.....	23
<i>Study Conduct</i>	23
CHAPTER IV	25
RESEARCH RESULTS	25
<i>Findings</i>	25
System Description	25
IIT: Description.....	29
IIT: Advantages.....	32
IIT: Disadvantages	33
SIT Technique: Description	35
SIT: Advantages.....	39
SIT: Disadvantages	40
WSIT: Description	41
WSIT: Advantages	44
WSIT: Disadvantages.....	45
User Manual.....	47
<i>Conclusions</i>	47
IIT	47
SIT	48
WSIT	49
<i>Recomendations</i>	50
CHAPTER V.....	53
RESEARCH IMPLICATIONS.....	53
<i>Organization Implementation</i>	53
Implementation Process	53
Implications If System Not Adopted	54
<i>Future Research</i>	55

CHAPTER VI	58
CONCLUSIONS.....	58
REFERENCES	60
APPENDIX 1.....	72
APPENDIX 2.....	73
APPENDIX 3.....	74
APPENDIX 4.....	75
APPENDIX 5.....	76
APPENDIX 6.....	77
APPENDIX 7.....	78
APPENDIX 8.....	79
APPENDIX 9.....	80
APPENDIX 10.....	81
APPENDIX 11.....	82
APPENDIX 12.....	83
APPENDIX 13.....	92
APPENDIX 14.....	93

LIST OF TABLES

	<u>PAGE</u>
1. Software Metrics Measured.....	19

LIST OF FIGURES

	<u>PAGE</u>
1. The Action Research Cycle	22
2. GreenEdmonton.ca Use Case Diagram.....	26
3. Architecture Of Three Different Integration Techniques.....	27
4. IIT Architecture.....	27
5. GreenEdmonton.ca PHP Source Code Directory And File Structure.....	28
6. Output of “View Events by Month” Use Case.....	30
7. GreenEdmonton.ca – Graphical User Interface.....	30
8. Example of UI Component Conflict	31
9. SIT Architecture	36
10. “Search Event” Graphical User Interface.....	37
11. Output of “Search Events” Use Case	39
12. WSIT Architecture.....	43

CHAPTER 1

INTRODUCTION

Statement Of The Problem

Integrating computer applications running on different platforms and/or written in different languages is a very important task in today's enterprise. PHP: Hypertext Preprocessor (PHP) is an emerging player in the field of web application computer languages, but its adoption in the enterprise is limited, in part due to integration issues.

Business Opportunity. PHP is the technological foundation for many Open Source Software (OSS) projects that have the potential to be customized and extended by organizations. Since most medium and large organizations have primarily Java Platform, Enterprise Edition (JEE) or Microsoft .NET (.NET) Information Technology (IT) resources, there are barriers to these organizations integrating their IT systems with these PHP projects. Developing an efficient, elegant mechanism for integrating PHP-based systems with JEE-based ones (this report will focus on JEE rather than .NET) is an opportunity for organizations to realize some important benefits that OSS projects offer.

Investigating the best methods for integrating PHP and JEE applications can help organizations maximally leverage PHP-based OSS projects as well as existing investments in JEE-centric human resources and systems. These integration methods can help organizations integrate their JEE systems with PHP ones. While the literature covers the high-level issues of integration patterns and architecture as well as more specific

issues such as integrating JEE and .NET applications, very little has been written about integrating PHP and JEE applications.

This thesis report will investigate various JEE-PHP integration techniques and critically analyze their pros and cons. It doing so, it will contribute to an under-documented area of literature that has the potential to help organizations better capitalize on the opportunity that PHP-JEE integration provides.

Technical Problem. The integration of units of functionality that execute in different application environments is a challenging task. The technique that one chooses when integrating the two units of functionality is crucial. It is important that the investment in leveraging the units of functionality in each of the systems is less than would have been invested in simply disposing of one of the systems, and rewriting its functionality in the native language and environment of the other.

This thesis report seeks to answer some research questions through the implementation of an existing JEE subsystem with Drupal, a PHP-based OSS Content Management System (CMS) (Drupal Association, n.d.a). The resulting system has been published as a live web application, www.greenEdmonton.ca (see “System Description”, p. 19).

Three different integration techniques were attempted. During the implementation of the system, answers for the following questions were sought for each technique:

1. What were the disadvantages/advantages of the integration technique for organizations considering its implementation?
2. Would the integration technique be more appropriate for small/medium applications versus large ones?

3. Which of the techniques required the most effort to implement?
4. Was there any opportunity to leverage the interface code/logic of the existing JEE system, or should just the business layer be leveraged?

Seeking the answers to these questions, through the implementation of a real-world system using existing OSS CMS software, enhanced the quality of this thesis report by providing realistic problems and obstacles to overcome.

Impact of the Problem/Opportunity. PHP is a popular programming language that has recently been making serious inroads into the realm of enterprise-scale computing. With enterprise-level software providers like International Business Machines Corporation (IBM) (LaMonica, 2005) and Microsoft Corporation (Kerner, 2006) introducing support for PHP, and with the availability of “tools to support the enterprise-grade servers that run [enterprise-grade] applications” (Wayner, 2005, para. 1), PHP has developed into a competitor to the JEE and .NET platforms in the enterprise sphere.

Because of PHP’s origin as an OSS programming language and its reputation as a platform that is easy to learn, OSS projects developed in PHP often have very large, active user communities supporting them. Many of these projects, if their functionality can be cheaply and efficiently leveraged through customization, extension, and/or integration, have a significant potential to add value to organizations of all sizes.

Potential upsides to using an existing OSS system as a starting or extension point for an internal IT project include a lack of startup cost (OSS is generally freely available), access to the internal source code of the system, and a wide variety of system extensions and plug-ins to choose from if the OSS community is active and widespread.

OSS CMSs provide an excellent opportunity for organizations to leverage existing functionality and code when implementing integration projects. These CMSs already have much of the “plumbing” code such as user subsystems, content presentation subsystems and content management (such as blogging, wiki, and file management) subsystems that many integration projects need. Of course, in order to leverage the code in an OSS CMS, developers and analysts must be able to efficiently hook into the CMS’ existing code base.

While there are certainly opportunities for organization to benefit from the adoption of open source *commodity* software such as operating systems, office productivity software, and database software, it is most likely that an organization will want to customize and extend OSS projects to develop *in-house* software. “In-house development - creating software to meet the specific needs of an organization ... results in the majority of software development activity” (Hart, 2001, para. 6), and it produces software that is customized and specific.

In-house software development in medium and large size organizations is often focused on business system integration. In fact, according to a 2004 survey of Chief Information Officers at Fortune 500 companies, “integration of business applications is the top challenge for companies hoping to improve their efficiency” (Lamonic, para. 5). Much of this integration is Information Portal Integration (IPI), the “aggregat[ion] of information from multiple sources into a single display to avoid having the user access multiple systems for information” (Hohpe and Woolf, 2003, p. 6). CMSs are an appropriate foundation for IPI projects.

When looking to solve an IT problem or fill a gap in an IT system, organizations can choose to buy a prebuilt solution from a commercial vendor, build the solution from scratch using in-house resources, or build a solution by customizing and/or extending an existing open source solution. There are risks with each of these approaches. The more that these risks can be mitigated for any given approach, the more attractive that approach becomes.

The extension and customization of PHP-based OSS projects introduce risks for organizations. “The risk with open source projects is that an IT department might lack the skills and the development capacity to tailor the project to meet the requirements and then support the customization going forward” (Guliani, 2005, p. 140). Therefore, if techniques for integrating and extending PHP-based systems are not well understood, risks for organizations that attempt to leverage PHP-based OSS projects are increased. The impact of these risks can be severe, from the widespread effect of bad architectural decisions to bugs introduced into systems by incorrect data mapping techniques.

Another risk that organizations face is that, without an understanding of the feasibility of leveraging PHP code, they may not realize the value-adding potential that PHP-based OSS projects have. This could cause organizations to make poor decisions such as deciding to build solutions from scratch that replicate functionality that already exists in one or more PHP-based OSS projects.

Potential Causes of the Problem/Opportunity. Due to reasons stated earlier, most of the most popular OSS CMSs are written in PHP. Of the five finalists for Packt Publishing’s 2006 Open Source CMS Award (Singh, 2006), four of them are written entirely in PHP. This factor presents a problem when considering the leveraging of OSS

CMSs by organizations, because their developer bases are significantly skewed towards having JEE and .NET skills. In fact, a 2006 Evans Data Corporation survey sites only C/C++, Visual Basic 6.0, .NET and Java (the “J” in JEE) in its list of programming languages used by systems developers, with PHP being relegated to the “other” category (2006).

If an organization is to successfully leverage an OSS CMS for an in-house application integration project, it must either acquire or train some PHP developer resources, or find a mechanism to integrate the PHP-based OSS CMS with existing and future JEE applications. Since acquiring or training PHP developers is mostly an organizational (versus purely technical) challenge, this thesis report focused on mechanisms to integrate PHP and JEE.

Integrating a JEE application with one written in PHP is a challenging task that requires attention at various levels. After all, the applications are coded using different programming languages and executing within different application environments that are possibly running on different operating systems. When performing JEE-PHP integration, developers face the challenges of, among other things, avoiding the introduction of brittleness to the application, ensuring that data types are properly and accurately mapped between JEE and PHP, and ensuring that the integration solution is performant.

Potential Solutions to the Problem/Opportunity. Three different integration techniques were attempted for this thesis report. They were chosen because they each offer different implementation complexity, benefits, and drawbacks.

The first technique was to put the PHP system and the JEE subsystem behind a common firewall. The PHP system integrated the JEE subsystem use cases into its

interface by implementing code that simulated the behavior of a client program (in this case, a Web browser). This client program code submitted input to and retrieved output from the JEE subsystem, customized the output so that it integrated with the PHP interface, and presented it to the client of the PHP system (the Web browser that was accessing the PHP system). In doing so, the PHP system passed security credentials to the JEE subsystem in order to access secure areas. The technique is referred to in this document as the Interface Integration Technique (IIT).

The second integration technique was based on an OSS project called the PHP/Java Bridge. PHP/Java Bridge is “an optimized, XML-based network protocol, which can be used to connect a native script engine, PHP, with a Java ... virtual machine” (PHP/Java Bridge Project Team, n.d.a, para. 1). A socket-based solution, PHP/Java Bridge allows a developer to directly access the Business Logic Layer (BLL) Java classes from a PHP environment. This technique was easily implemented and very performant (since it uses a proprietary communication protocol, it does not bog down the network with as much overhead data as is needed by standardized protocols). The technique is referred to in this document as the Socket-Based Integration Technique (SIT).

The third integration technique that was attempted for this thesis report is one that makes use of Web services standards such as Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) to expose business logic functionality written in Java to the PHP environment. This technique has widespread support within industry and academia. Although their performance suffers somewhat due to their high level of abstraction and dependence on the text-based Extensible Markup Language (XML) (Vaughan-Nichols, 2002), Web services are being widely deployed as Enterprise

Application Integration (EAI) solutions. They solve many integration issues because “SOAP is ... a platform-neutral and language-neutral choice ... [that] has widespread industry support” (Daconta, Smith & Obrst, 2003, chap. 4, section 2). While possibly being the most complex of the three integration techniques to implement and deploy, Web services offered the highest degree of adherence to industry best practices. The technique is referred to in this document as the Web Services Integration Technique (WSIT).

CHAPTER 2

REVIEW OF RELATED LITERATURE

This thesis report is about IT system integration. More specifically, it was an investigation of conceptual, technical and architectural challenges faced by developers attempting to integrate PHP systems with JEE systems. Some of the current literature as it applies to this thesis report is outlined in this section.

Integration Project Types

Every integration project involves some form of information sharing between disparate applications. This information sharing, especially at the enterprise level, can take many different forms. Hohpe and Woolf identify five types of intra-organizational integration projects: Information Portals (IPs), Data Replication (DR), Shared Business Functions (SBFs), Service-Oriented Architectures (SOAs), and Distributed Business Processes (DBPs) (2003, p. 6). The type of integration attempted depends on an organization's short- and long-term goals.

Hohpe and Woolf's concept of DBPs are described by Loke & Ling as "a set of activities or tasks which need to be executed in some controlled order to realize a business purpose" and which can be performed "by moving to and interacting with resources[,] possibly over an Intranet" (2000, p. 460). DBP integration projects would be implemented primarily by large organizations that have many different systems. These complex projects would involve the bringing together of input and business logic from two or more different systems to create a value-adding automation of a business process that spanned

the different systems. Due mainly to scope concerns, DPB issues were not considered during the course of this thesis report.

Historically, many application integration projects have enabled greater interactivity between systems by extracting and processing data from the database end of application architectures, and then applying business and presentation logic to add integrating value. Akshinthala Kumar refers to this technique as "extract, transform, and load" (ETL) (2005, para. 1). ETL is "the creation of [a] common logical representation and the relationships between [this common logical representation] and the sets of input and transformed data" (Kumar, 2005, para. 10). This approach is still commonly applied today, albeit with more modern techniques, and often incorporating Web services into the implementation. With its focus on data, ETL would be categorized by Hohpe and Woolf as DR (2003, p. 6). It was not considered for this thesis report due to its failure to maximally leverage existing business logic, and its omission of the presentation logic layer (PLL) that PHP is best suited for.

This thesis report investigated techniques that would be used in integration projects that Hohpe and Woolf describe as SBF projects, IP Integration (IPI) projects and SOA projects (2003, p. 6). These techniques are of special interest because OSS CMSs and PHP are particularly well-positioned to provide the basis for solutions them.

An IPI approach, one that "achieve[s] integration by bringing together information from many different systems within a single user interface" (Linthicum 2003, Chapter 1), is suitable for this thesis report because IPI is a very common challenge to organizations. Also, IPI is an ideal problem for PHP- JEE integration to solve.

PHP is renowned as “a fast and powerful scripting language designed specifically for web applications” (Lahav, 2002, p. 273) (web applications comprise the interface of most portals), and the PHP-based OSS CMSs contain a lot of portal-like functionality. Furthermore, maximum value can be extracted from an application by leveraging all of its functionality, from PLL to persistence layer. Therefore, how to capture all of the functionality in a PHP-based OSS CMS like Drupal by using it as the basis of a portal that integrates existing JEE functionality is a worthwhile topic of research for this thesis report.

“Portals aggregate information from multiple sources to present a unified interface to the end user” (Gold-Bernstein & Ruh, 2004, Section 10.3.6). Almost every computer user has, at some point in their lives, used computer systems that were deployed by the same organization, but that required using separate interfaces and providing security credentials separately to each interface. Virtually combining two or more organizational applications into one through an IPI project increases user productivity by reducing the number of times the user has to supply security credentials, by exposing systems that were previously not available to the user, and by “grouping information logically to facilitate navigation” (Jacoby, 2005, p. 38).

This paper will also investigate techniques that would be used by SBF and SOA projects. SBF projects involve “enabling legacy systems for integration within an enterprise [by] making the data and transactions enabled across the enterprise instead of being limited to the specific vertical application that the legacy code was developed for” (Arsanjani, Hailpern, Martin & Tarr, 2002, p. 5). This thesis report will share business functions of an existing JEE system in two ways: By exposing business logic using Web services (which are a major component of SOA projects) and by exposing business logic

with PHP/Java Bridge (PHP/Java Bridge Project Team, n.d.a). These techniques will be a microcosm of enterprise-level SBF projects that use various techniques for leveraging existing business logic functionality by exposing it to other applications within the enterprise.

In the context of integrating PHP and JEE applications, one of the techniques that this thesis report will investigate is the “integration of components by combining their presentation front-ends, rather than their application logic or data” (Daniel et al., 2006, p. 1). Much groundwork has been laid in regards to creating standardized, reusable User Interface (UI) components. The Java Portlet Specification (JSR 168), for example, allows developers “to create pluggable components that run on any compliant, JEE portal server” (Klaene, 2004, para. 2). The same type of UI standardization is available for .NET under the “Web Parts Framework” (Walther, 2005). Thus, a piece of system functionality that has a UI built following the JSR 168 or Web Parts Framework standards will most likely be able to be fully leveraged (from the front end, all the way to the back end) as a component in a portal other than the one that it was initially deployed in. However, the system functionality would not be easy to leverage from a non-JEE (or non-.NET) system. The project completed for this paper will attempt to leverage UI logic that does not follow any UI specification (such as JSR 168) from an existing JEE system by incorporating it into the native UI of Drupal, the PHP-based OSS CMS that will form the basis of greenEdmonton.ca.

Best Practices

Enterprise Integration has been a major industry push for the past several years. As such, best practices are well established. This thesis report will analyze the efficiency

and utility of the PHP-JEE integration techniques in part by discussing how closely they adhere to the industry-recognized best practices of loose coupling and adherence to standard interfaces and open systems.

For business “processes that span multiple business units or enterprises”, that is, for processes that require integration solutions, “there appears to be a consensus that ... a loosely coupled architecture is required because loose coupling is seen as helping to reduce the overall complexity and dependencies. Using loose coupling makes the application landscape more agile, enables quicker change, and reduces risk” (Krafzig, 2004, Section 3.5). The lessons that developers learned in the 1980s and 1990s when their tightly-coupled objects made desktop applications brittle and difficult to maintain apply even more so when integrating disparate computer applications. The loose coupling of interacting components, whether they are distributed components objects within the same application environment, is a widely-accepted industry best practice.

“The core principle behind loose coupling is to reduce the assumptions two parties (components, applications, services, programs, users) make about each other when they exchange information” (Hohpe & Woolf, 2003, Chapter 1, Section 5). The more assumptions that the two parties make, the more maintenance that is involved when one of the parties is changed. Therefore, maintenance and upgrades become more difficult as the coupling between parties (the number of assumptions made) becomes tighter. This thesis report will critique its PHP-JEE integration techniques in part by analyzing how tightly they couple the PHP and JEE applications.

One of the concepts that Land & Crnkovic list as important to system integration is adherence to “Standard Interfaces and Open Systems” (2004, section 1.2). More

specifically, the best practice to apply to system integration is the creation of “a set of components with interface specifications fully defined, available to the public, maintained according to group consensus, and in which the implementations of components are conformant to the specification” (Land & Crnkovic, 2004, section 4, para. 1). This helps architects avoid vendor lock-in and maximizes the potential for future integration projects to hook to the systems being integrated. This paper will analyze how closely its attempted integration techniques promote the systems being integrated as being open, with standard interfaces.

REVIEW OF ORGANIZATION DOCUMENTS

Ecology Systems Information Society

The Edmonton-based Ecology Systems Information Society (ESIS) agreed to be the sponsor for the project. ESIS was incorporated in 1993 (incorporation #505725192).

One of the aims of the society is to act as an information clearinghouse about environmental and social justice issues like transportation (its relationships to land use, sprawl, equitability, fossil fuel, pollution, safety, etc.), climate change, biotechnology, energy and water conservation and efficiency, pesticides, and urban agriculture. Information is gathered and shared with the public using the media and a variety of avenues (local community newsletters, Environmental Non-government Organizations, an information centre in a high traffic location, event hosting, etc.).

ESIS was founded with the belief that if citizens were given information, they would be motivated to change their lifestyles, which in turn would change the course of society. To further this goal, ESIS provides a library of books, clipping files, and videos to the public. Also, its coordinator Michael Kalmanovitch sends out a weekly e-newsletter called the Activist Agenda (AA) (Kalmanovitch, n.d.). Creating the AA involves the gathering and organizing of information about Edmonton-based events, jobs, and other items of interest to the activist community. The process of receiving, processing and formatting the AA's information and maintaining its e-mail list is time-consuming.

In conjunction with the author of this report, ESIS developed the concept for a web-based IT system that will further the goals of ESIS by implementing new ways to disseminate information as well as automate most of the business processes involved with

creating the AA. Dubbed greenEdmonton.ca, the system will bring ESIS' original goal of being an information clearinghouse into the Internet age.

GreenEdmonton.ca Concept

GreenEdmonton.ca will be a community-driven online portal offering local sustainable lifestyle and consumer solutions for Edmontonians. The heart of greenEdmonton.ca will be a searchable database of well-researched consumer information focusing on sustainable and environmentally-friendly lifestyle choices. In order to facilitate Edmontonians in making these choices, greenEdmonton.ca will provide links between general consumer information and local sources for “green” products. Not only will greenEdmonton.ca tell consumers which washing machine to buy, for example, it will also tell them where they can buy it. Additionally, greenEdmonton.ca taps into Edmonton's collective environmental knowledge by encouraging visitors to submit their own tips and articles on how to live “green” in Edmonton. GreenEdmonton.ca will also feature other community-driven online “green” lifestyle tools such as (possibly) a ride/carpool board, the AA newsletter, a local events calendar (integrated with AA), and a swap (freeshare) board. A greenEdmonton.ca Use Case Diagram can be found in Figure 2.

GreenEdmonton.ca Technical Considerations

When the author began considering the technical options for creating greenEdmonton.ca, it became apparent that much of the required functionality was already available in various OSS CMS projects. The best of these projects, with the richest features and the most active user communities have been built using PHP. The author's primary area of expertise is building web applications using JEE technologies

(Hibernate, Struts, Tomcat, etc.). Because of this, a decision needed to be made between extending a JEE-based OSS CMS that was of suboptimal quality and extending one of the best OSS CMSs available (a PHP-based one) by integrating it with custom JEE code.

Recognizing that this problem is one faced by countless organizations when they contemplate some form of in-house IT system development, ESIS and the author decided to propose the building of greenEdmonton.ca as a mechanism by which to research PHP-JEE integration techniques. The results of the research are presented in this thesis report.

CHAPTER III

RESEARCH METHODOLOGY

Research Methods

The methodology for this thesis report consisted of the project implementation and the subsequent analysis of the efficacy of the implementation techniques used, through reflection and comparisons with industry and academic literature. This style of research is a type of action research.

Action research is a participatory, democratic process concerned with developing practical knowing in the pursuit of worthwhile human purposes, grounded in a participatory worldview ... [It] seeks to bring together action and reflection, theory and practice, in participation with others, in the pursuit of practical solutions (Reason & Bradbury, 2001, p.1).

The author used action research combined with qualitative analysis as opposed to a more quantitative approach such as an experimental model using software metrics. The qualitative analysis involved comparing the different techniques from an implementer's perspective, using industry best practices as reference points.

Quantitative, Metrics-based Approach. Software metrics have limited utility for small systems, especially those that execute in diverse operating environments. At 7911 original lines of source code (LOC), the AA is even below the 10,000 to 50,000 lines threshold that Grable, Jernigan, Pogue, & Dale reserves for "small applications" (1999). Furthermore, a LOC in one operating environment, such as PHP, can be quite different

from another in Java, just as a LOC “in an assembly language is not comparable in effort, functionality, or complexity to an LOC in a high-level language” (Fenton & Neil, 1999, p.150). So, although the metrics of “time to code” (TTC) and LOC were measured during the course of this project, their validity and worth are in question.

The author kept track of the TTC, measured in hours, for each of the integration techniques implemented (See Table 1), and found the metric lacking. It was difficult to know when to begin timing each implementation, because they each required some level of configuration, and the line between configuration and programming is sometimes not easy to determine. Furthermore, the steep learning curve inherent in the first time an implementation technique is attempted makes the TTC for each integration technique misleading for the purposes of predicting the development time for future coding efforts. Finally, although there was naturally a correlation between the hours programmed and the author’s subjective opinion on how difficult each technique was, this correlation contributed little to the conclusions of the thesis.

Integration Technique	Programming Hours	Lines of Code
IIT	32	51
SIT	39	74
WSIT	41	244

Table 1 – Software Metrics Utilized

The LOC metric was also tracked. As with the TTC metric, it was difficult to make valid-seeming conclusions based on the LOC. Others have made similar observations:

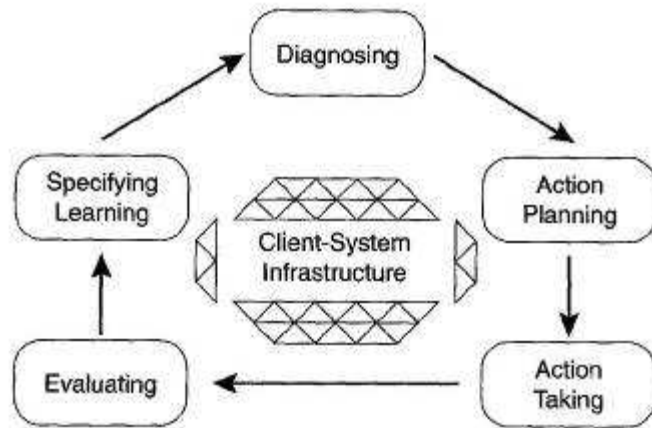
This term [LOC] is highly ambiguous and is used for many different counting conventions. The most common variance concerns whether physical lines of logical statements comprise the basic elements of the metrics. Note that for some modern programming languages that use button controls, neither physical lines nor logical statements are relevant. (Jones, 1997, p.333)

Like languages with button controls, the basic element of the LOC with web-based interfaces is very difficult to establish. Programmers will break up HTML output to the browser over many different lines in order to keep their code readable, so deciding what qualifies as a LOC is a subjective process. Furthermore, with the prominent role played by configuration files (such as XML files), which give directions to the program but are not executed directly, the level of ambiguity is even higher. So, comparing lines of code written in XML, JSP, and Java files in the AA to each other as well as to those written in Drupal seemed like an exercise doomed to reach invalid conclusions.

For a metrics-based research approach to be successful, a research project needs to be designed to be measurable from its inception. For any given metric, a small project does not represent a large enough sample size to enable valid conclusions to be reached for that metric. Therefore, factors that might be confounding must be eliminated by means other than a large sample size. For example, this study could have been designed so that a particular use case was exactly replicated using the three different integration techniques. Run in isolation on the same server, these techniques could have been measured for performance (measured in milliseconds of execution time) with a significant degree of accuracy, because most if not all confounding factors would have been eliminated due to the low degree of variability between the situations being measured.

Due to the difficulties inherent in measuring differences between the attempted integration techniques, the author chose to reject the use of metrics altogether in reaching conclusions for this thesis. This decision allowed for action research to be performed, research with an “expected benefit for both researcher and organization” (Baskerville & Wood-Harper, 1996, p.239). It allowed for the avoidance of creating a system that was measurable but not useful. So, a qualitative analysis, with comparisons to research and industry best practices, was undertaken.

Action Research and Qualitative Analysis. Action research can take many different forms. The type of action research used for this thesis report roughly followed Susman and Evered’s description (as cited in Baskerville & Wood-Harper, 1996, p.238), which identifies five phases: Diagnosing, action planning, action taking, evaluating, and specifying learning (See Figure 1). At every phase, the corresponding descriptive action



The action research cycle (Susman, 1983)

Figure 1

(diagnosing, action planning, etc.) was taken with regards to both the research question and the organizational problem.

“Action research attempts to link theory and practice, thinking and doing, achieving both practical and research objectives” (Susman, 1983). With action research, “the researcher is actively involved, with expected benefit for both researcher and organization” (Baskerville & Wood-Harper, 1996, p.239). Therefore, action research is the ideal methodology for this thesis report. Furthermore, qualitative analysis based on comparisons to the academic literature and industry best practices made the thesis’ findings more useful than they would have been had a more quantitative approach been taken. These approaches will contribute to accurate qualitative analysis that will serve as a useful addition to the literature on systems integration.

Development and Deployment Environment and Tools

Software. The greenEdmonton.ca web system was developed using 100% OSS. The AA subsystem was coded in the Java programming language using the Eclipse 3.2 Integrated Development Environment (IDE) (Eclipse Foundation, n.d.). The PLL was developed using the Struts 1.3.0 web application framework (Apache Software Foundation, n.d.c.) and Java Server Pages (JSP) 2.1 (Sun Microsystems, n.d.a.). The AA subsystem DAL logic was developed using the Hibernate persistence layer framework (JBoss, Inc., n.d.) All of the code for the subsystem was executed and tested using the Apache Tomcat 5.5.9 web server (Apache Software Foundation, n.d.b.) running in a Java Virtual Machine (JVM) provided by the Java Development Kit 1.5.0_06 (Sun Microsystems, n.d.b.)

The PHP portal into which the AA subsystem was integrated was customized from the Drupal OSS CMS (Drupal Association, n.d.a). PHP code was altered and added using the Eclipse IDE, and the code was developed and tested using the PHP 5.2.0 Scripting Engine installed on Apache Hypertext Transfer Protocol (HTTP) Server 2.2.3 (Apache Software Foundation, n.d.a.).

Hardware. The project was developed on a Dell Inspiron 6400 laptop (Dell, n.d.) with 1.5 GB of RAM and a dual core Intel processor (Intel, n.d.). GreenEdmonton.ca is deployed on a seven-year-old PC running the Ubuntu Linux operating system (Canonical Ltd, n.d.). It is connected to the Internet via a standard business DSL line.

Study Conduct

For this thesis report, ESIS provided the client system infrastructure. The diagnosing of the organizational problem (phase 1 of the action research cycle) involved

the realization that current information dissemination techniques were inefficient and insufficient (hence the need for greenEdmonton.ca). Phase 2 involved the creation of a use case diagram and a problem description document for greenEdmonton.ca, and the formulation of the research problem that included a plan to integrate an OSS PHP system with an existing JEE subsystem. Phase 3 involved the actual integration of Drupal (the PHP-based OSS system) with the JEE subsystem, which entailed the completion of more than a dozen new use cases.

Phase 4 will involve "a determination of whether the theoretical effects of the action were realized, and whether these effects relieved the problems" (Baskerville & Wood-Harper, 1996, p. 238). This evaluation process will be conducted by Michael Kalmanovitch of ESIS and the author of this report, and it will focus on how successful greenEdmonton.ca is at furthering ESIS' goals. Finally, Phase 5 of the action research included reflection and qualitative analysis of how the attempted research techniques solved the technical problems in question. Chapters IV, V, and VI of this thesis report are the outcomes of this "Specifying Learning" phase of the action research.

CHAPTER IV

RESEARCH RESULTS

Findings

System Description. GreenEdmonton.ca has approximately 50 use cases, about 20 of which exist as functioning units in the AA subsystem (implemented using JEE). Of the remaining use cases, approximately 25 of them exist in a version of Drupal that has been downloaded, customized, and upgraded with free plug-in modules. Those cases that were not implemented in either the system (Drupal) or the subsystem (AA) were written as custom-built Drupal modules.

GreenEdmonton.ca is divided into four subsystems: The User, Blog, Freeshare, and AA subsystems (See Figure 2). As a general-purpose CMS, Drupal offers a basic install that includes a User subsystem and a Blog subsystem. In order to serve its primary purpose of being an information clearinghouse, the greenEdmonton.ca UI features the Blog subsystem predominantly in its user interface. Since the web site will continually be refreshed by new blog postings, having the Blog subsystem occupy a central area of the UI will keep the user community interested, and attract new users.

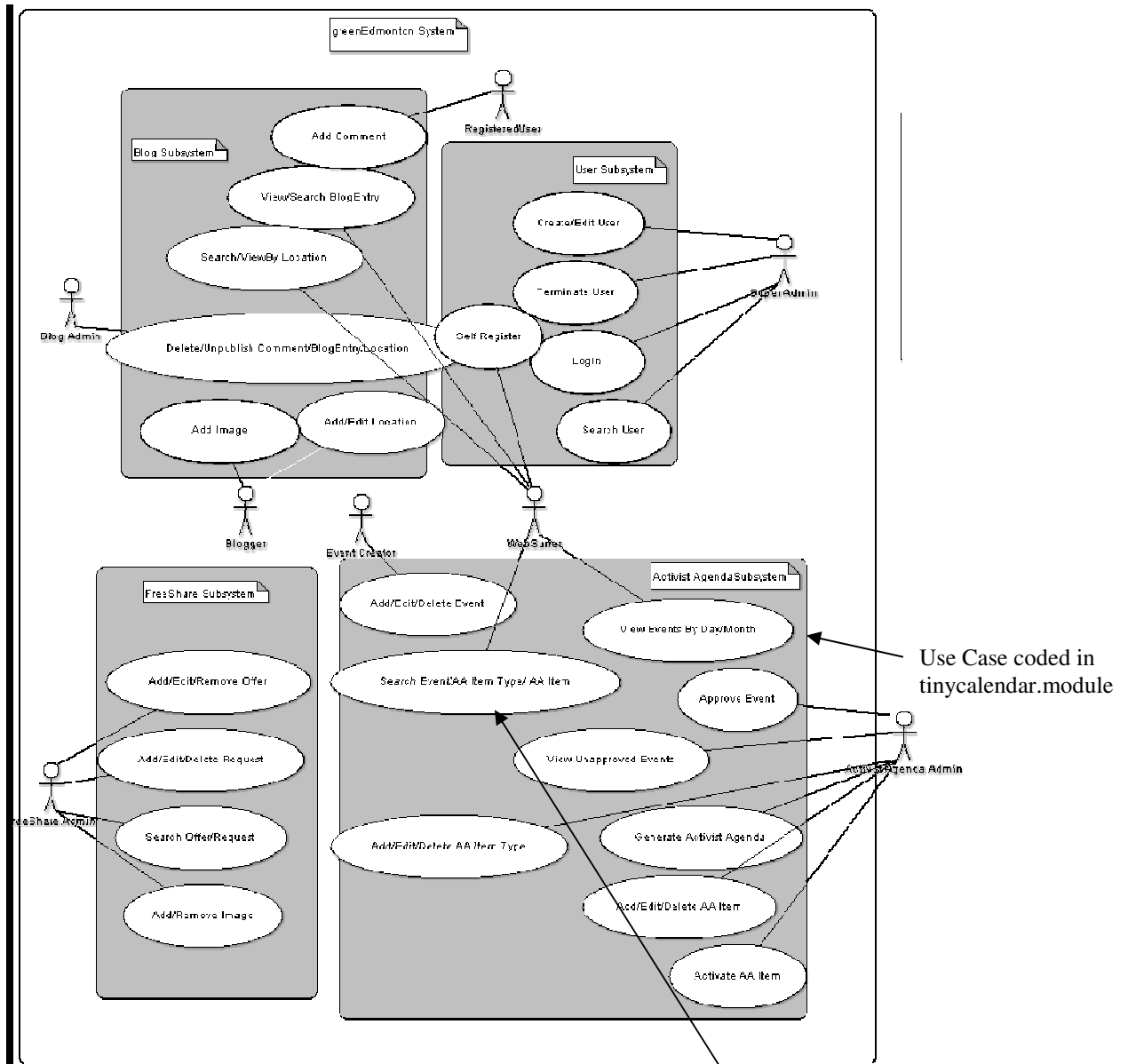


Figure 2 – GreenEdmonton.ca Use Case Diagram

Use Case coded in activistagenda.module

The User subsystem was already implemented in both the AA and Drupal, so special consideration had to be taken with regard to integrating the implementations. For increased, more specialized functionality, the Drupal community has developed dozens of add-on modules that can be downloaded and installed.

Daniel et al. (2006) describe three different integration techniques, each of which integrates applications at a different layer (see Figure 3). Within a layered architecture,

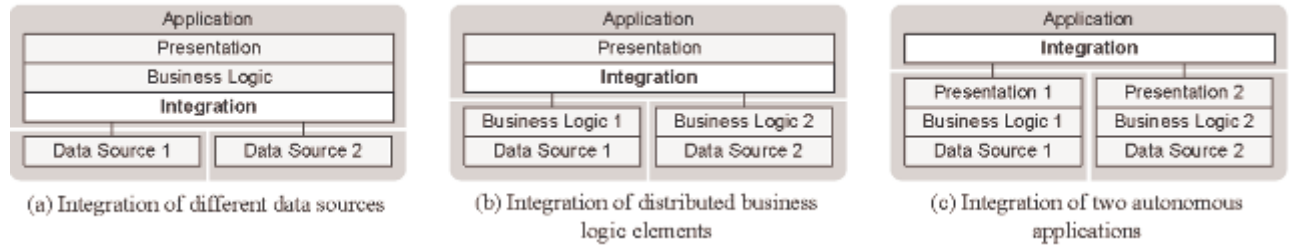


Figure 3 (Daniel et al., 2006)

Daniel et al. introduce the concept of a separate integration layer that communicates with and combines either the persistence layers, the BLLs, or the PLLs of two applications (a, b, and c, respectively, of Figure 3).

In this research project, due to Drupal's relatively unstructured architecture, the interface integration logic resides in Drupal's PLL (which also contains some business logic). It communicates with the appropriate layer of the AA subsystem depending on which of the techniques is being implemented (for example, see Figure 4).

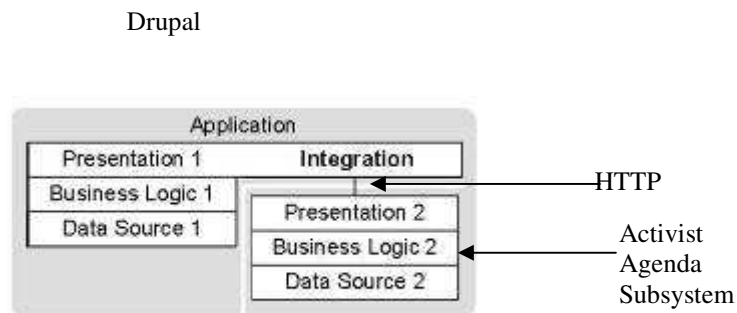


Figure 4 – Architecture of the IIT

Since the integration code is embedded in Drupal's PLL code, the integration programmer must have an in-depth knowledge of the Drupal engine in order

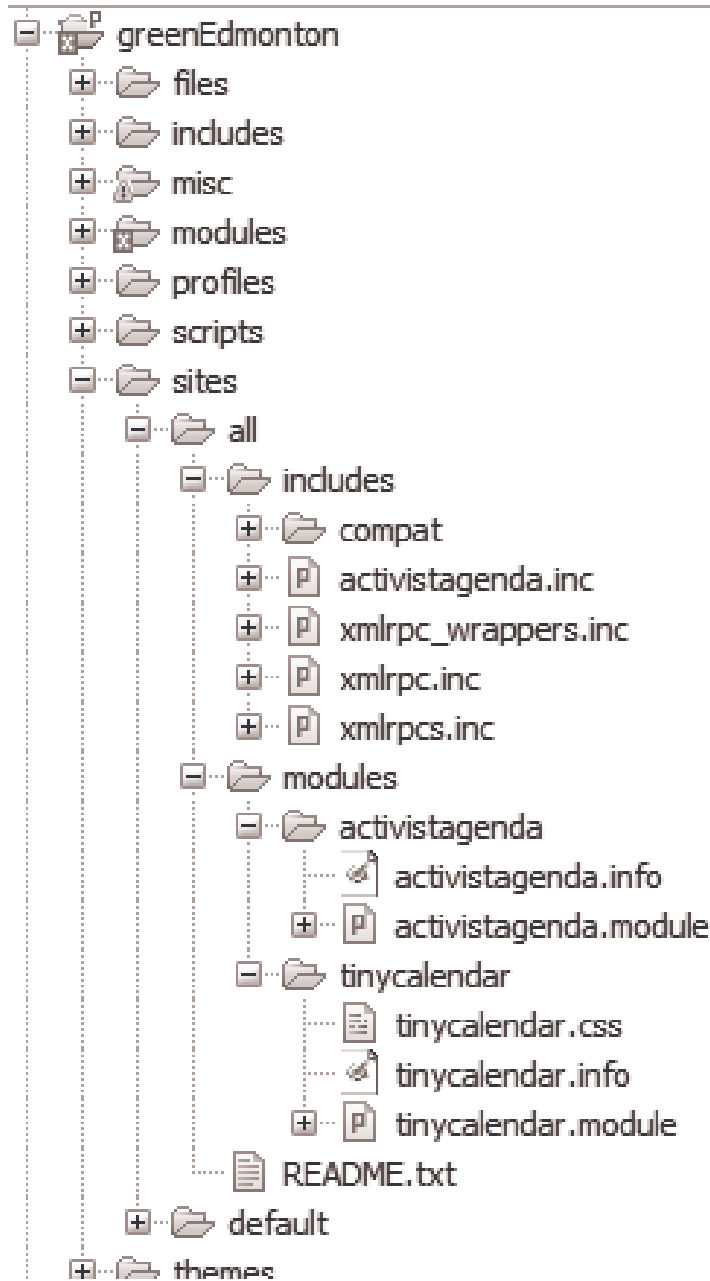


Figure 5 – greenEdmonton.ca directory and file structure

to hook the integration code into it properly. The developer must also ensure that the integration code is not coupled with other Drupal modules in order to ensure that these modules can be upgraded in the future without overwriting the integration code.

Therefore, the integration code was placed in the "sites/all/modules" directory of the base Drupal installation (see Figure 5). With a few commands from Drupal's administrator

interface, Drupal recognizes the modules from the "sites/all/modules" directory and executes the code contained in its functions.

Although PHP has object-oriented capabilities (Suraski, 2002), much PHP code is written using a structured programming style. The PHP code in Drupal is no exception, with stand-alone functions being the common method of encapsulating programmatic behavior. Further encapsulation was achieved by organizing the functions in PHP "include" files (files with the ".inc" suffix). The common, reusable functions for the AA integration were placed in the file `activistagenda.inc` (see Figure 5).

IIT: Description. The first of the integration techniques used for the project integrated the AA into the larger system by programmatically interacting directly with the AA's interface. Otherwise known as presentation integration or "screen scraping", this technique allows for all of the logic that is written in the application, including the presentation, business, and persistence layer logic, to be leveraged. "The widespread trend of equipping applications with Web-based interfaces has revived interest in using Presentation Integration as a vital integration approach" (Microsoft, n.d.). The HTTP's request-response cycle, as well as the ease with which HTML can be programmatically parsed, make an HTML-based interface such as the AA subsystem's a good candidate for interface integration.

The IIT is implemented in Drupal's PLL, and it communicates with the AA subsystem's presentation via HTTP requests and responses (see Figure 4).

The use case "View Events by Month" of the AA subsystem (see "Use Case coded in `tinycalendar.module`", Figure 2) was developed and implemented using JEE. It displays one month of events at a time (see Figure 6), and provides links for the user to

move to "next" and "previous" months as well as to view the events for a day, if they exist (in Figure 6, events exist for February 22).



Figure 6 – the output of “View Events by Month”

The code to integrate "View Events by Month" into Drupal is placed in `tinycalendar.module` (see Figure 5), and then `tinycalendar` is registered as a Drupal add-on module by the Drupal administrator. From the point of view of Drupal's PLL, `tinycalendar` is a "block", a small piece of the UI that “can be placed in any sidebar/region” (Drupal Association, n.d.b). `Tinycalendar` is visible on every page that the user visits (see Figure 7, right side of web page). When a user visits



Figure 7 – GUI - GreenEdmonton.ca

the `greenEdmonton.ca` home page, Drupal calls the “`tinycalendar_block`” function from `tinycalendar.module` to render the `tinycalendar` output (seen in Figures 6 and 7).

Drupal executes within a PHP environment, which executes as a module of the Apache HTTP server. The AA subsystem, on the other hand, executes within the Tomcat

web server, which executes within a JVM. When Drupal makes a call to the `tinycalendar_block` function, then, the function's code communicates with the Tomcat server, and feeds the response to the Drupal engine. A new HTTP request object is created by passing the Uniform Resource Locator (URL) of the AA subsystem's hook to the "View Events by Month" use case to an `HttpRequest` constructor (see Appendix 1, line 4).

If the HTTP request is successful (as indicated by an HTTP response code of 200, see Appendix 1, line 7), then the HTML that the AA subsystem outputs to display its tiny calendar is available to the PHP code in the "\$request" object. This is raw HTML that still needs to be integrated into the HTML generated by the Drupal engine. The first time that this code was attempted, the resulting Drupal display had some unexpected green and gray bars going through the tiny calendar (see Figure 8). This happened because when integrating two interfaces "code isolation is not guaranteed ... and conflicts among UI components may occur" (Daniel et al., 2006).

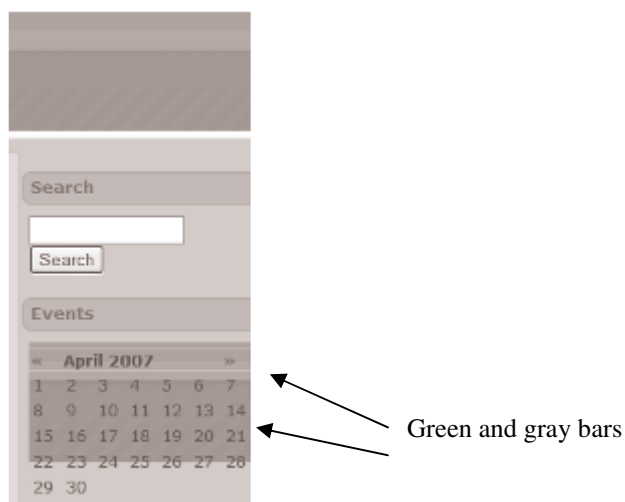


Figure 8 – Example of UI Component Conflict

The output scraped from the AA subsystem consisted of an entire HTML page. After the "body" tag, the following code appeared: `<div id="header">`. Because Drupal links to a Cascading Style Sheet that adds style information to "div" tags that have the ID "header" (see Appendix 2, lines 1-6), an unwanted image of a gray and green bar was displayed as the background of the tinycalendar output. This code collision was remedied by stripping out the offending "`<div>`" tag (with a call to "`strip_markup_from_JEE_response`". See Appendix 1, Line 8) as well as other HTML elements received from the AA subsystem that interfered with a clean display of the tinycalendar in Drupal.

The output from the AA subsystem contains elements that link back to the AA subsystem. These links needed to be transformed so that they linked to Drupal. A local URL path was created in Drupal, and the offending links were replaced with links to the Drupal URL paths (see Appendix 1, lines 13 and 18). Then, code was written that was triggered by the Drupal URL paths being requested. This code intercepts the request, redirects it to the Tomcat server in which the AA subsystem is executing, and displays the AA subsystem's output to the Drupal user.

IIT Advantages. The IIT, although it is often referred to in derogatory terms (Comella-Dorda, Wallnau, Seacord, & Robert, 2000), has its advantages.

Firstly, once an understanding of the Drupal architecture was achieved (this understanding would be required regardless of the integration technique), the author was able to immediately begin integrating the two applications. That is, there were no external systems to configure, deploy, or install. Furthermore, the code needed to communicate between the systems was quite simple, and made use of existing PHP

objects. This simplicity was aided by the fact that communication was using HTTP, which is a ubiquitous standard. This simplicity of initial effort, along with the ease with which the integration environment is configured (that is, just start coding directly in PHP), could be a significant advantage if time were of the essence.

Secondly, a big advantage of this technique was that the UI logic and layout embodied in the AA subsystem was fully leveraged. Because of this, a minimal amount of PLL needed to be written when integrating the applications via their interfaces. The data that was received from the AA subsystem was laid out and organized with logic that was already implemented in JEE. The effort required to strip out unwanted markup code that conflicted or otherwise disrupted Drupal's UI was quite minimal compared to the effort that would have been required had the integration code simply received streams of unformatted data.

Finally, since no new point of entry to the AA subsystem was created, the resources that had been expended to make the AA subsystem secure were fully leveraged when using the IIT. The PHP code to simulate a browser logging in to the AA subsystem was nontrivial, with special attention required to perfectly mimicking the HTTP headers transferred between Drupal and the AA subsystem during the login process. Once the login simulation was accomplished, though, no new security analysis needed to be performed on the subsystem, because the AA subsystem's previously existing, fully tested credential-granting process was used.

IIT: Disadvantages. Although still very widely used, IITs have a bad reputation. Shields & Owen complain that “screen scraping...tends to be slow, cumbersome and difficult to maintain” (2003) and another technology writer states that “screen

scraping...makes for tedious programming" (Dietzen, 2004). Obviously, there are some significant disadvantages to integrating applications through their interfaces.

One aspect of the applied IIT that soon emerged was the extremely tight coupling that existed between the Drupal system and the AA subsystem. If the AA subsystem's interface was upgraded or changed in any way, there would be a strong possibility that the integration of the two systems would be compromised.

For example, the ActionServlet that is the Struts controller's engine (the AA subsystem is programmed using the Struts application framework, see Appendix 14) knows when it is being requested, by matching the extension of URLs submitted to the server in which it is running (Apache Software Foundation, n.d.d). The specific extension that the ActionServlet matches is configured in its web application's web.xml configuration file (see Appendix 14). The AA subsystem's ActionServlet is configured to watch for the pattern "*.do" at the end of URLs that are submitted to its web server (see Appendix 3). The extension that it watches for could effortlessly be changed to any other pattern by simply changing what is found between the <url-pattern> tags in web.xml (see Appendix 3). The integration code in Drupal that implements the "Add Event" use case (see Figure 5), however, would cease to function properly if this change was made. The "*.do" pattern is hard coded into the integration code, because "*.do" is part of the interface for the AA subsystem. In order to ensure that the AA subsystem's "Add Event" form submits to Drupal instead of back to the AA subsystem, the URL that it submits to is changed from one with the "*.do" pattern to one that submits to Drupal (see Appendix 4, Line 14). This tightly couples the system with the integrated subsystem, thereby introducing brittleness should the subsystem's interface be tampered with in any way.

SIT Technique: Description. The second of the integration techniques used for the project was one that integrated the AA into the larger system by instantiating Java objects from the BLL of the AA using an open source bridging technology called the PHP/Java Bridge (PHP/Java Bridge Project Team, n.d.a).

The PHP/Java Bridge was installed by incorporating its backend into the Tomcat web application in which the AA executes, and its front end in the PHP environment of the Drupal system. Java objects could then be instantiated from Drupal using PHP functions through the Java servlet PhpJavaServlet (PHP/Java Bridge Project Team, n.d.b). PhpJavaServlet “handles PUT requests [a type of HTTP request] and then re-directs to a private (socket- or pipe-) communication channel” (PHP/Java Bridge Project Team, n.d.b). Once the private communication channel is established, PHP/Java Bridge accepts object instantiation and method call commands from the PHP client, and enacts those commands from the JVM in which it resides. It then passes the resulting Java object(s) and/or primitive data type(s) back to the PHP client using “an optimized, XML-based network protocol” (PHP/Java Bridge Project Team, n.d.a). The PHP/Java Bridge client, in turn, makes the Java object(s) or primitive(s) available to the PHP environment through a PHP variable.

As with the IIT, the integration code resides in Drupal's PLL. Using this technique, PLL constructs such as forms, links, and interface instructions are implemented using Drupal's built-in PLL hooks. Once input for a use case has been collected from the user and validated, integration-layer code instantiates an object from the AA subsystem's BLL. In effect, this instantiation attempt involves making a HTTP request to the PhpJavaServlet that is running in Tomcat. The PhpJavaServlet redirects the

PHP execution environment to an awaiting socket, where it sends a stream representing the resulting Java class, object, or primitive type encoded in a custom-designed XML-based protocol. In this way, user input from Drupal is sent to the AA subsystem's BLL, and the results of that input are displayed using PHP code and Drupal's built-in PLL hooks (see Figure 5).

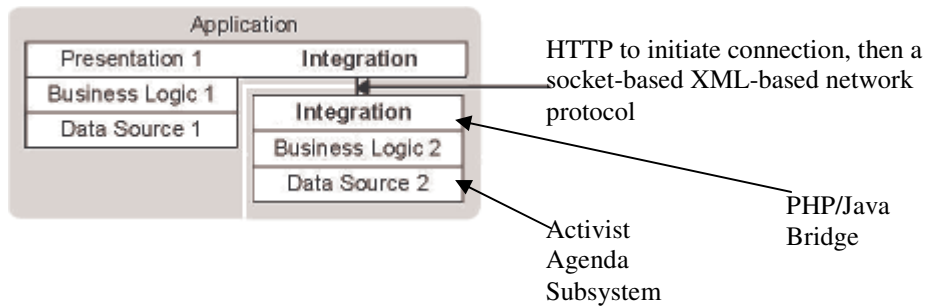


Figure 9 - Architecture of the SIT

The use case "Search Event" of the AA subsystem (see Figure 5) was fully developed and implemented using JEE. The AA subsystem's PLL code for the use case was not reused for integration because its reliance on the Struts framework in which it was developed makes it inconvenient to access using the PHP/Java Bridge. Instead, the PLL was reimplemented in PHP/Drupal. The user is presented with a form containing the searchable fields of the "Event" entity (see Appendix 13, and Figure 10).

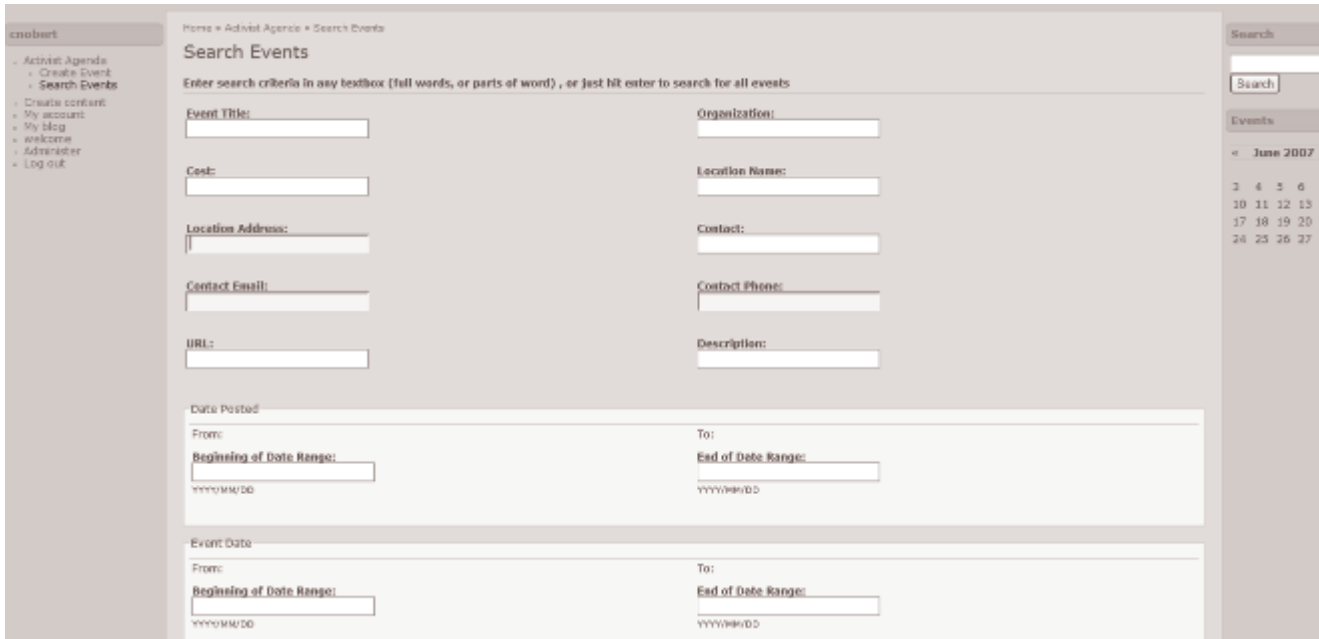


Figure 10 – “Search Event” GUI

Once the user submits the values that she wants to search on, Drupal executes the “event_search_form_submit” function (see Appendix 5) from `activistagenda.module` (see figure 5). This function instantiates a local instance of the `ca.activistAgenda.managers.EventManager` class using a Java object constructor provided by the PHP/Java Bridge (see Appendix 5, line 4). It then makes a call to `ca.activistAgenda.managers.EventManager`’s “search” method (see Appendix 5, line 11), passing in the values that were submitted by the user through Drupal (contained in the `$form_values` variable).

At this point in the integration process, the point at which the components in the system begin to interact with those in the subsystem, the developer must be careful to avoid what Yakimovich, Travassos, & Basili call 2nd-order semantic-pragmatic incompatibilities. These incompatibilities are “caused by interaction of two components” when “different assumptions [are made] between [them]” (1999, p. 4). For example, when

initially developing the “event_search_form_submit” function (see Appendix 5), an “argument type mismatch” exception was thrown from the JEE system. Upon investigation, it became clear that two components, one from the PHP system and one from the Java system, while working correctly in their respective environments, make different, clashing assumptions.

The PHP function `strtotime`, which converts a properly formatted string to a date/time, “returns a timestamp on success, FALSE otherwise” (PHP Group, n.d.a). So, when the user enters nothing in a text box that should contain a date (see Figure 6), passing the contents of the empty text box to `strtotime` returns a Boolean value of false. However, when this Boolean value is passed to the Java environment (see Appendix 5, line 10), an “argument type mismatch” exception is thrown, because the conversion method in the Java environment returns a null value rather than a Boolean value, given an empty string. Therefore, `ca.activistAgenda.managers.EventManager.search()` expects to be passed either a valid Date or null. In order to mitigate this 2-order semantic-pragmatic incompatibility, the variables storing the contents of the date-related text boxes are first initialized to null (see Appendix 5, line 5), and the return value of `strtotime` is only stored in them if they contain a valid date (see Appendix 5, line 6).

After the form has been accepted and validated by Drupal and its contents have been submitted to the AA subsystem, Drupal redirects the user's browser to an internal URL that triggers `activistagenda_eventSearch` (see Appendix 6). This function retrieves the Java List (Sun Microsystems, n.d.c) that was previously stored in the session (see Appendix 5, line 29) and iterates through it by calling the same methods as one would in Java, but using PHP syntax (see Appendix 6, lines 5-8). The results are then displayed to

the user by Drupal (see Figure 11).

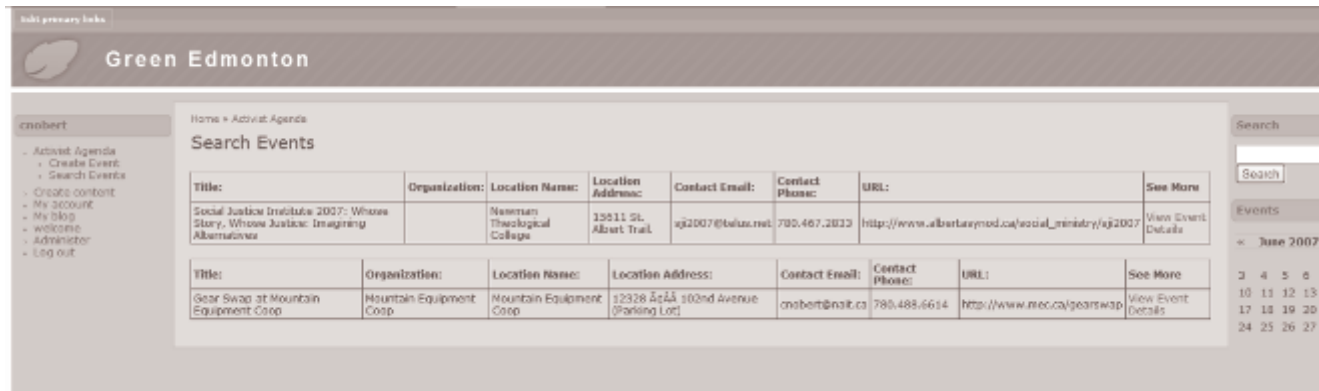


Figure 11 – Output of “Search Events” Use Case

SIT: Advantages. The SIT provided by the PHP/Java Bridge has some distinct advantages over other methods.

The socket-based mechanism of the PHP/Java Bridge implementation allows the Drupal system to directly access the BLL of the AA subsystem (see figure 5). This direct access reduces the coupling between system and subsystem by eliminating the assumption by the Drupal system that the AA subsystem’s PLL will remain unchanged. Since "the less two [entities] know about each other, the more loosely coupled they are to each other" (Basham, Bates, & Sierra, 2004, p. 712), the fact that the Drupal system knows nothing about the AA subsystem’s PLL loosens the coupling between system and subsystem, thereby making the application less brittle and in closer adherence to industry best practices. It also gives the organization responsible for the AA subsystem the option of continuing to support and improve its PLL, while at the same time making the subsystem’s business logic available to another system.

The relatively simple configuration and setup of the PHP/Java Bridge is an advantage. The back end of the Bridge was installed directly in the web application in

which the AA subsystem executes. Therefore, no new servers, or even Web applications within an existing server, needed to be installed and maintained. This reduces the cost of integration in person hours and hardware, and reduces the dependencies between components in the architecture of the given organization.

SIT: Disadvantages. The SIT provided by the PHP/Java Bridge has disadvantages related to security, Java/PHP object compatibility, and lack of adherence to standard interfaces.

The security provided by the AA subsystem is implemented using JDBC realms within a Tomcat server installation (Apache Software Foundation, n.d.e). Because PHP/Java Bridge enables the direct instantiation of Java objects, one would need access to the environment in which Tomcat was running, in order to access the JDBC Realm-based objects that implement security for that server. Since the back end of the PHP/Java Bridge runs within a Java Web Application, it can only provide access to objects that are available to servlets executing within that Application. The Tomcat developers could not provide access to the Tomcat server objects, because then individual servlets could be written in a way that subverted the execution environment in which the servlets were running. Therefore, there is no way for the PHP/Java Bridge to know which areas of the web application are available to which level of user. This is a distinct disadvantage, because the security that is built into the AA subsystem must be enforced by some mechanism other than the PHP/Java Bridge. This inability to leverage the existing security-based logic in the AA subsystem introduces complexity into the integration process.

The Java objects that are made available by the PHP/Java Bridge client are not true PHP objects, but thin wrappers around Java objects that are accessed using PHP syntax. This is a disadvantage because it makes for unpredictable interaction between the PHP "objects" that represent Java objects and the true PHP functions and objects that are available in the PHP environment. For example, when the PLL for the use case "View Event Details" (see Figure 2) was coded, the author attempted to convert the "date posted" field from the "event" object into a string by using the built-in PHP function `strftime` (PHP Group, n.d.b) (see Appendix 7, line 4). Due to incompatibilities in how Java and PHP represent dates, the resulting outputted string was always "Mon, Dec 22 2031", regardless of what the actual date was. In order to solve the problem, the author converted the day to using another Java object (see Appendix 8, lines 1 and 4). In this instance, the "\$event" object was not a true PHP object, but simply a thin wrapper around a Java object that was passed from the AA subsystem to the Drupal system via the PHP/Java Bridge. This fact introduced more complexity than if the bridging solution properly converted the Java object to a PHP object.

Finally, the PHP/Java Bridge integration mechanism has the significant disadvantage of not adhering to well accepted standards. It is implemented using a unique protocol, and, as its name implies, it does not offer integration between many types of platforms. Therefore, the resources expended in implementing this socket-based solution would need to be replicated if the AA subsystem were to be integrated into a non-PHP based system.

WSIT: Description. The third of the integration techniques used for the project integrated the AA into Drupal by instantiating Java objects from the BLL of the AA using

the Apache eXtensible Interaction System (Axis) SOAP engine and SOAP Remote Procedure Calls (RPC) conventions (Apache Software Foundation, n.d.f.).

Axis was installed by installing the AxisServlet servlet in Tomcat and exposing it using a public URL. The AxisServlet acts as the Tomcat server's public access point to the Axis SOAP engine, which is instantiated and executed as needed within the servlet environment. Once a WSDL descriptor file was written to communicate which Java classes should be accessed by the Web service invocation (i.e. the call to AxisServlet), a deployment operation was executed to make the Axis SOAP engine aware of the instructions in the WSDL file. In PHP, the PHP SOAP functions were enabled by making a simple change to the php.ini file (University of Washington, n.d.) of the PHP installation. Finally, the Web service could be invoked in PHP (i.e. by the Web service's client) by passing the URL of the service's WSDL file to the PHP SOAP function's "SoapClient" constructor.

This integration technique is a small example of what might appear in a SOA. As such, it exposes functionality from the subsystem's BLL (as opposed to its PLL or DAL) (See Figure 12). The PHP SOAP functions communicate with the AxisServlet using the ubiquitous HTTP, and once the business functionality is exposed on the server side, any client that adheres to the SOAP RPC specifications should be able to access it.

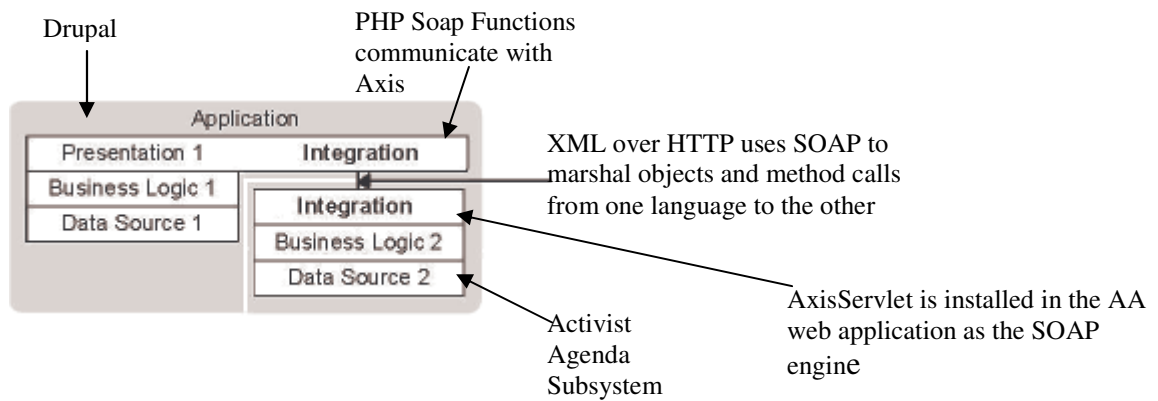


Figure 12 – Architecture of the WSIT

The use case "View Events by Month" of the AA subsystem (see Figure 2) was integrated into the system using this technique. As with the PHP/Java Bridge technique, only the functionality from the AA's BLL (and therefore also its DAL, which is accessed by the BLL) was integrated into the system. So, the PLL for the use case was rewritten in PHP.

Implementing the server-side portion of the Web service, that is, exposing the `ca.activistAgenda.managers.EventManager.getEventsByMonth()` method as a Web service, was surprisingly difficult. Interestingly, there is "nothing in the SOAP specifications which covers complex objects" (Apache Software Foundation, n.d.g). Therefore, the `java.util.List` object, which represents a collection of Event objects, needed to be replaced by a Java array. This was accomplished by writing a wrapper class, `EventManagerSoapBindingImpl`, which loops through the `java.util.List` object and loads its contents into an array. Furthermore, the Event objects, each containing a User object and a collection of EventDateTIme objects, needed to be mapped to XML Schema types

in EventManager.wsdl, and then registered with Axis by telling it “which Java classes map to which XML Schema types” (Apache Software Foundation, n.d.h) in deploy.wsdd (the Web Service Deployment Descriptor file).

The use case is invoked when the user clicks on a URL titled "View [month's] Events" (for example, "View December's Events"). This invokes the `activistagenda_eventsview_by_month` in Drupal's `activistagenda` module (see Appendix 9).

The WSIT allows for very simple, pure-PHP programming. Once the input is in the XML data type “xsd:dateTime” format, and the remote method called (see Appendix 9, lines 8 and 10), the `$events` variable contains an array of PHP structures. These can be looped through using native PHP constructs (see Appendix 9, line 22) and parsed using PHP functions such as `strtotime` and `date` (see Appendix 9, lines 25-27).

WSIT: Advantages. There are two major advantages of the Web Services technique: true conversion of data types and objects to the language of the client and very high levels of interoperability between disparate operating environments.

The SOAP engines involved in integrating the AA with the greater system truly convert the objects being passed around from Java to PHP and back. Therefore, when the PHP client receives its array of PHP structures, the developers using the array to write interface layer code can fully leverage their PHP expertise. This "true" conversion of programming constructs from one language to another reduces the risks of errors being introduced into the code due to assumptions either PHP or Java makes about the data types with which it is interacting. The PHP/Java Bridge solution, where the objects received by the remote procedure call are actually Java objects wrapped thinly in PHP,

could introduce subtle, difficult-to-debug errors as Java objects are manipulated by PHP code.

"Together with XML, Web service protocol standards provide an industry-wide means to realize interoperability and platform-neutral communications" (Hewlett-Packard Company, 2001), and "Gartner Dataquest research shows an increasingly widespread use of Web Services standards" (Cantara, 2005, para. 6). Due to this widespread adoption of these standards, investment in exposing business functionality through Web services is very easy to leverage at a future date. The initial extra effort (over, for example, the ITT) required to integrate with Web services can easily pay off, as the business logic that is exposed can be integrated in as many different systems as one wishes, regardless of what technology they are implemented in.

WSIT: Disadvantages. The WSIT suffers from the disadvantages of a tedious data mapping process, security issues, and the need to write wrapper classes for methods that return collections of objects.

The WSDL file describes a published Web service to potential Web service clients. Among other things, this file contains a detailed mapping of the underlying programming language's data types to the SOAP RPC standards. The process of creating this mapping for every Web service that is published is an extra layer of coding and maintenance work that developers must concern themselves with. The process involves listing every element of a Java data type and, if it is a primitive type, explicitly linking it to its SOAP RPC equivalent. If the element is a complex data type, it is mapped to its data type mapping definition (see Appendix 10). While there are tools sold by vendors such as Altova (n.d.) that make the process more efficient, the fact that the extra mapping takes

place will inevitably require more effort as the Web service and its underlying business logic implementation evolve and are maintained.

The Web services technique suffers from the same security-based deficiency as the PHP/Java Bridge solution: it has no way of leveraging the roles-based, JDBC Realm-implemented security that is built into the AA subsystem. Whereas the screen scraping technique directly accesses the PLL of the subsystem, Web services expose business objects, which are implemented in the BLL. Security needs to be re-implemented, then, for the WSIT. This is a redundant process, as security has generally already been worked out and implemented in the original subsystem.

There is “nothing in the SOAP specifications which covers complex objects. The most reliable way to send aggregate objects is to use arrays” (Apache Software Foundation, n.d.g) This fact introduces the disadvantage of needing to write a facade method in the underlying programming language if a Web service method returns something other than a primitive data type or an array of primitives. This means using the underlying programming language to iterate through the non-array collection-type object and create an array that can be returned by the Web service (see Appendix 11). This process adds no value to the underlying business functionality. It is purely to accommodate the inability of SOAP to represent collections of complex data types. IBM even recommends Java developers to “stick with the simple data types as much as possible. Totally avoid those fancy complex [sic] types such as ArrayList, Tree, and even the common Hashtable” (Ye, Jan 2005, Conclusion) in order to make the deployment of Web Services easier. This need to send only simple arrays across the network when using Web services is detrimental because it causes extra work

for the developer. Also, it could encourage developers to "code to Web services", thereby making design decisions to make deploying Web services easier, rather than to solve the business problem that they are writing the code for in the first place.

User Manual. See Appendix 12 for the AA user manual as implemented on greenEdmonton.ca.

Conclusions

IIT. The IIT can be very useful in integrating Java subsystems into PHP applications if the conditions are right. These conditions include a subsystem with a PLL that will never be upgraded or changed again, and relatively simple user interfaces for both the system and subsystem.

The IIT causes the system interface to be very tightly coupled with the subsystem interface. Screen scraping, as interface integration is often referred to, "is notoriously fragile to even minor layout changes" (Wright, 2002, p. 50). Therefore, this technique would be a poor choice to integrate a subsystem that still had an active user interface, because any changes to the subsystem's interface could have functional consequences for the system into which it was integrated. The only appropriate subsystem for this technique is one that would never again be directly accessed through its native user interface.

With HTML-based systems such as the ones integrated in this project, there is a significant risk of UI component conflict. Given an HTML page, for example,

...what the user sees as 'blue text' in [the] rendered page may be blue for many reasons: because it is a hyperlink; because it is contained in a FONT tag; because it has a CSS style attribute; because it matched by a CSS stylesheet rule; or

because its color attribute was set by some JavaScript code. Because of this the need to examine and understand the HTML source is a roadblock that discourages spur-of-the-moment innovation. Web automation must be done at the level of rendered pages. (Miller, 2003, p. 4)

Combining HTML pages to make a coherent interface, then, becomes unacceptably complex if the pages are non-trivial in nature. JavaScript elements of a UI have an especially high chance of conflicting with the UI integrated from the subsystem. Therefore, the IIT is suitable only for applications that have relatively simple interfaces that do not rely heavily (as many new Asynchronous JavaScript and XML- (AJAX-) based interfaces do) on complex client-side scripting.

If the two systems have simple HTML-based interfaces and the subsystem's interface will not be subjected to future changes, interface integration can be very effective at fully leveraging all of the investment in the subsystem. Furthermore, an integration project can be ramped up very quickly due to the simplicity of programmatically querying the subsystem for its HTML-based PLL hooks. It appears that the IIT -- the much denigrated "screen scraping" integration technique -- will continue to be useful to organizations under certain conditions.

SIT. The PHP/Java Bridge has the potential to be a good integration technology in some situations, especially for small and medium-sized organizations or large organizations with homogeneous application environments.

Many small and medium-sized organizations will only ever need a small number of computer systems. For them, the PHP/Java Bridge offers a quick installation with low overhead on the Java server where the subsystem resides. Like Web services, the

PHP/Java Bridge allows a system to integrate the BLL of the subsystem while still allowing the subsystem's interface to evolve. Unlike Web services though, the SIT does not require an extra layer of data mapping code to be written on the subsystem's Java server. It also promises low network activity overhead, because it does not need to convert objects to the text-based XML before passing them over the network.

The PHP/Java Bridge is much less suitable for large organizations with heterogeneous application environments. These organizations have many different computer systems, and they have no way of predicting which ones will need to be integrated with others in the future. Therefore, interoperability is a key asset to obtain when integrating applications. Since PHP/Java Bridge offers interoperability with a limited number of application environments, the organization could be missing out on a chance to leverage its investment in the future if it chooses to integrate using the PHP/Java Bridge instead of Web services.

WSIT. SOA adoptions, which are primarily implemented using web services, "... [are] now reaching ubiquitous implementation in the United States, regardless of size or vertical industry designation (Grid Today, 2005, para. 1). Interoperability between heterogeneous operating systems and execution environments is absolutely vital for these organizations to leverage past IT investments.

Although exposing business functionality using Web services requires considerably more effort than doing so using the PHP/Java Bridge, its legacy includes the ability for other applications of all types to easily integrate with the exposed business functionality. This is a very valuable legacy because there could be a need to integrate the application with yet another application or subsystem in the future. This future

integration would be relatively cheap due to the preexistence of the Web services hooks, and the original investment would be fully leveraged.

Large organizations have a dynamic and evolving human resource skill set. The "true" integration that Web services offer – that is, the true conversion of primitive and complex data types from one programming language to the other – is important to these organizations because it allows them to leverage the specialized skills that their architects and programmers possess. When integrating a Java subsystem into a PHP system, for example, the organization can assign different human resources respectively to PHP programming, Web services data mapping, and Java programming. The ability for these human resources to specialize in a technology area should allow them to be more productive than if they needed to program across programming language and execution environment boundaries.

Recommendations

“Software architects undertake a number of crucial tasks during the design of integrated enterprise applications ... amongst these are...selecting suitable integration technologies that can fulfill the application requirements” (Gorton & Liu, 2005, p. 726). Once these technologies are selected, they lock the integration project into a set of constraints and opportunities that is usually for all practical purposes irreversible. Therefore, software architects and other decision-makers should weigh all of their options carefully, and with as much information at their disposal as possible.

Decision-makers should strongly consider the opportunities provided by OSS CMSs when considering IPI projects. The best of the OSS CMSs contain much of the functionality required by many IPI projects. Furthermore, one of the major risks of their

adoption -- their language of implementation -- can be mitigated by adopting one of the integration techniques discussed in this project thesis report. The investment of leveraging the open source code in these CMSs should be weighed against the investment in rewriting the code in a format and language more familiar to the organization and purchasing a proprietary CMS.

When making decisions about which technologies and techniques to adopt for integration projects, decision-makers should keep in mind that not all technologies are suitable for all situations. While it may be tempting to simply adopt the technology that was used on the last project or the one that is trending at the time that the decision is made, decision-makers need to remember that every integration project is unique. Using Web services, for example, to integrate a rarely used, dead-end application written in an obscure language is probably not a good choice. For such a project, a simpler, less trendy option such as user interface integration would probably be cheaper and quicker. After all, “the undeniable benefit of screen scraping is that the legacy application does not require any changes whatsoever” (Noffsinger, Niedbalski, Blanks & Emmart, 1998, p. 82). Nearly every option, as concluded in this report, is appropriate in some situations.

“To maximize IT investment value, a good manager must size up the relevant risks and proactively build flexibility into an investment” (Benaroch, Jeffery, Kauffman & Shah, 2007, p. 110). Therefore, decision-makers should be very careful before foregoing the use of Web services to integrate applications. While they may certainly be the best choice in a given situation, many non-web service based solutions to integration involve relinquishing the opportunity to leverage the investment in the future. So, there should be a high level of certainty that an integration project is too simple to justify the

added overhead of Web services or that the subsystem to be integrated will definitely never be required by another application before dismissing Web services as part of an integration solution.

CHAPTER V

RESEARCH IMPLICATIONS

Organization Implementation

ESIS is an organization run primarily by Michael Kalmanovitch and staff members of Earth's General Store, of which he is the sole proprietor. The benefits that the greenEdmonton.ca web site will provide once implemented are a reduction of workload by Kalmanovitch in creating and maintaining the AA e-newsletter and the streamlining of ESIS' role as an information clearinghouse by allowing event and item submitters direct access to the AA.

Implementation Process. In order to become accepted as the web portal of choice for Edmonton-based environmentalists, greenEdmonton.ca is being implemented in stages.

In September, 2007, the blogging subsystem was made publicly available on the Internet. Once a substantial amount of content has been submitted in order to establish the legitimacy of the web site, ESIS will begin a recruitment campaign to attract Edmonton-based bloggers who are leaders in green lifestyles and environmental activism.

In November, 2007, the AA subsystem was made publicly available on the Internet. For the time being, the author will mirror Kalmanovitch's AA newsletter on greenEdmonton.ca. This will allow Kalmanovitch to slowly transition to the new system. Once Kalmanovitch is entering all of the events and items directly into the greenEdmonton.ca system, Kalmanovitch will begin directing event and items submitters

directly to the web site. Once the majority of submitters are entering their information directly into greenEdmonton.ca, Kalmanovitch's workload will be greatly reduced.

Implications If System Not Adopted. Currently, the AA is a labor-intensive newsletter whose contents are all funneled through Kalmanovitch. This has the consequences of sapping the time of ESIS' primary human resource and missing out on significant value that greenEdmonton.ca could bring to the AA's content.

Interested parties in the Edmonton area currently e-mail Kalmanovitch their events and items, and he formats and edits them into the AA newsletter. This process is very time-consuming for Kalmanovitch, the principal employee of ESIS. It takes five to ten hours per week of his time. If greenEdmonton.ca, with its automatic formatting and direct entry by the submitters of events and items, is not adopted, the AA will continue to consume a large portion of ESIS' human resources. Furthermore, there will remain a direct relationship with the AA's success (measured through more event and item postings) and the amount of time it takes to manage the newsletter.

Successful internet applications are ones whose content communities of users can shape, maintain, and influence. No one person or small group of people has the energy or resources to compete against a self-motivated user community. In his much-read Web 2.0 manifesto, Tim O'Reilly discusses how Amazon.com,

...like competitors such as Barnesandnoble.com, [bought] its original database came from ISBN registry provider R.R. Bowker. But ... [Amazon] harnessed their users to annotate the data, such that after ten years, Amazon, not Bowker, is the primary source for bibliographic data on books, a reference source for scholars and librarians as well as consumers. (2005, section 3, para. 6)

ESIS stands the same risk as Bowker if it continues to rely on a single employee to filter and update its AA newsletter. By allowing the Edmonton-based community of activists and change agents to directly enter items and events into the AA, ESIS will be leveraging an engaged user community to better serve its role of an information clearinghouse about environmental and social justice issues.

Future Research

This thesis project report gives a qualitative analysis of different technological solutions to the problem of integrating Java- and PHP-based computer systems. A further discussion could be had about making EAI architecture and technology decisions in a more formal manner that includes an evaluation of the technological expertise that an organization possesses and the pros and cons of the various Java/PHP interoperability technologies.

“At the beginning of [an EAI project], software engineers need to conduct a trade-off study to evaluate the costs of the various interoperability solutions” (Reyes, Espino, Mohan & Nadkar, 2003, p. 1). A significant factor in such a trade-off study is what Kauffman and Li call “IT Switching Costs”. These costs refer “... to the costs incurred for a user to switch from one technology to another” (2004), and Li and Johnson identify them as 50% of the decision making factors in their framework for strategic IT investment evaluation (2002, p. 38). The ability to which an organization can leverage OSS CMSs and the suitability of different Java/PHP interoperability technologies can have a significant effect on IT switching costs (by enabling switching to a computer language that integrates well, or enabling the avoidance of switching altogether). Because of the significant influence of IT switching costs, research into a decision making matrix

that included the architectural platforms involved, the relative complexity of the IT project, the organizational expertise, and the different integration technologies would be valuable for architects, engineers and managers making decisions about integrating computer systems.

For some EAI projects, the performance of a subsystem that needs to be integrated becomes a significant factor. When researching communication between distributed Java applications, Juric, Kezmah, Hericko, Rozman, and Vezocnik reported that Java Remote Method Invocation (RMI), a native mechanism by which Java applications can integrate with one another over a network, "...offers an order of magnitude better performance than other alternatives, being at least 8.5 times faster than the second alternative – web services" (2004, p. 65). Given such a large potential difference between Web Services and other, less standardized integration technologies, similar research could be performed comparing the performance of the Java/PHP Bridge and Web Services. This would further enrich the decision-making information needed to choose how to integrate PHP and Java applications.

Part of this report is based on the premise that technology barriers play a role in discouraging organizations from adopting OSS CMSs in their IPI, SBF and SOA projects. A recent study involving over 200 interviews of corporate decision makers reported that 45% of surveyed companies using CMSs had connected some of their internal data to the CMS (Stahl & Mass, 2003, p. 7). A similar study could be made that asked organizations what factors had been included in deciding which CMS to adopt and how the CMSs had been integrated with their existing systems. Such a study would help to determine to what

extent and why PHP, the dominant implementation language of OSS CMSs, has been involved in integration projects and adopted by organizations.

CHAPTER VI

CONCLUSIONS

Integration of disparate software systems will continue to be a top priority for organizations. OSS CMSs provide an opportunity for organizations to cheaply and efficiently integrate legacy systems into new interfaces. A significant barrier to adoption of the best supported OMS CMSs, the PHP environment in which they are coded and executed, can be overcome by adopting appropriate integration techniques.

For most systems and organizations, integrating systems by exposing their functionality with Web services is an optimal solution. The potential to leverage the investment made in creating and publishing the Web services outweighs the advantages of other integration techniques in most cases. Since PHP offers Advanced Programming Interfaces that comply fully with Web services standards, the large amount of robust, useful functionality of available in OSS PHP systems (especially CMSs) can be leveraged by integrating them with legacy systems using Web services.

For very small and end-of-life legacy systems, IPI techniques (screen scraping) and specialized techniques such as the PHP/Java Bridge can offer organizations attractive options for short, cheaply-implemented integration projects. Careful consideration needs to be taken, though, as such techniques could force project rewrites and redundant effort if the importance and/or durability of the legacy systems is underestimated, and they need to be integrated again in the future.

The future should see further inroads for PHP in medium and large enterprises. These inroads will benefit by well thought-out and designed integration techniques, most of which will include the use of Web services.

REFERENCES

Altova, Inc. (n.d.). MapForce – *Graphical Data Mapping, Conversion, and Integration Tool*. Retrieved November 23, 2007 from

http://www.altova.com/products/mapforce/data_mapping.html

Athabasca University. (n.d.a). *Athabasca University Calendar 2006\07, Master of Science - Information Systems Homepage*. Retrieved March 9, 2007 from

http://www.athabascau.ca/calendar/grad/info_systems_01.html

Athabasca University. (n.d.b). *Master of Science - Information Systems Project Handbook COMP 697-COMP 699*. Retrieved March 7, 2007 from

http://scis.athabascau.ca/html/courses/comp697_698_699/ThesisProjectHandbook_V2005.pdf

Athabasca University. (n.d.c). *Computer Science (COMP) 697-699 Integration Project Syllabus Homepage*. Retrieved March 7, 2007 from

http://scis.athabascau.ca/graduate/comp697_698_699syllabus.htm

Arsanjani, A., Hailpern, B., Martin, J., & Tarr, P.L. (June 2002). Web Services: Promises and Compromises. *IBM Research Report RC22494*. Retrieved March 6, 2007 from

[http://domino.watson.ibm.com/library/cyberdig.nsf/papers/D5D90B0D8408E04885256BE400518B2A/\\$File/RC22494.pdf](http://domino.watson.ibm.com/library/cyberdig.nsf/papers/D5D90B0D8408E04885256BE400518B2A/$File/RC22494.pdf)

Apache Software Foundation. (n.d.a.). *Apache HTTP Server Project* . Retrieved February 27, 2007 from <http://httpd.apache.org/>

Apache Software Foundation. (n.d.b.). *Apache Tomcat Homepage*. Retrieved April 27, 2007 from <http://tomcat.apache.org/>

Apache Software Foundation. (n.d.c.). *Apache Struts Homepage*. Retrieved April 27, from <http://struts.apache.org/>

Apache Software Foundation. (n.d.d.). Configure the ActionServlet Mapping. *Apache Struts User Manual*. Section 5.4.2. Retrieved May 11, 2007 from http://struts.apache.org/1.2.9/userGuide/configuration.html#controller_config

Apache Software Foundation. (n.d.e.). Realm Configuration HOW-TO. *The Apache Jakarta Tomcat 5 Servlet/JSP Container*. Retrieved June 8, 2007 from <http://tomcat.apache.org/tomcat-5.0-doc/realm-howto.html>

Apache Software Foundation. (n.d.f.). Service Styles - RPC, Document, Wrapped, and Message. *Axis User's Guide*. Retrieved October 29, 2007 from <http://ws.apache.org/axis/java/user-guide.html#ServiceStylesRPCDocumentWrappedAndMessage>

Apache Software Foundation. (n.d.g.). What Axis can send via SOAP with restricted Interoperability. *Axis User's Guide*. Retrieved October 30, 2007 from <http://ws.apache.org/axis/java/user-guide.html#WhatAxisCanSendViaSOAPWithRestrictedInteroperability>

Apache Software Foundation. (n.d.h.). Encoding Your Beans - the BeanSerializer. *Axis User's Guide*. Retrieved October 30, 2007 from <http://ws.apache.org/axis/java/user-guide.html#EncodingYourBeansTheBeanSerializer>

- Basham, B., Bates, B., & Sierra, K. (Aug 2004). *Head First Servlets and JSP*. Sebastopol, CA : O'Reilly Media, Inc.
- Baskerville, R., & Wood-Harper, A. (Sep 1996). A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11, 3, pp. 235-246.
- Benaroch, M, Jeffery, M, Kauffman, R, & Shah, S. (2007). Option-Based Risk Management: A Field Study of Sequential Information Technology Investment Decisions. *Journal of Management Information Systems*, 24, 2, pp. 103-140.
- Canonical Ltd. (n.d.). *Ubuntu Homepage*. Retrieved May 1, 2007 from <http://www.ubuntu.com/>
- Cantara, Michele. (Nov 2005). Market Focus: Trends and Forecast for IT Professional Services for Web Services and SOA, 2005-2008. *Business Integration Journal*. Nov/Dec 2005
- Comella-Dorda, S., Wallnau, K., Seacord, R.C., & Robert, J. (2000). Proceedings from the International Conference on Software Maintenance: *A survey of black-box modernization approaches for information systems*. San Jose, CA. Retrieved May 4, 2007 from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=883039
- Daconta, M. C., Smith, K. T., & Obrst, L. J. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, New York, NY: John Wiley & Sons, Inc.
- Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., & Saint-Paul, R. (2006) Understanding UI Integration: A survey of problems, technologies, and opportunities. Technical Report DIT-06-064, Informatica e Telecomunicazioni,

University of Trento. Retrieved March 6, 2007 from

http://www.google.com/url?sa=t&ct=res&cd=1&url=http%3A%2F%2Fwww.cse.unsw.edu.au%2F~jyu%2Fiee-ic%2FUI_Integration.pdf&ei=6djtRcCdHaDagwOw48y2CA&usg=__TukjmXwZDpIjeUgCZQ4MvNEw6eY=&sig2=swRAJEfVr36Wm43w7QnEMQ

Dell Inc. (n.d.). *Inspiron 6400* . Retrieved May 1, 2007 from

http://www.dell.com/content/products/features.aspx/inspn_6400?c=us&cs=04&l=en&s=bsd

Dietzen, Scott. (Jan 2004). Standards-Based Integration: The Impact of Web Services and

JEE. *Dev2Dev: By Developers, For Developers*. Retrieved May 11, 2007 from

<http://dev2dev.bea.com/pub/a/2004/01/324.html>

Drupal Association. (n.d.a). *Drupal Official Website*. Retrieved March 1, 2007 from

<http://drupal.org/>

Drupal Association. (n.d.b). *Block Views*. Retrieved May 4, 2007 from

<http://drupal.org/node/54443>

Eclipse Foundation. (n.d.). *Eclipse Official Website*. Retrieved April 27, 2007 from

<http://www.eclipse.org/>

Evans Data Corporation. (2006). EMEA Development Survey, Winter 2006. Retrieved February 20, 2007 from

http://www.evansdata.com/n2/surveys/emea_toc_06_2.shtml

Fenton, N., & Neil, M. (July 1999) Software metrics: successes, failures and new directions. *Journal of Systems and Software*. 47, 2-3, pp.149-157.

- Gold-Bernstein, B., Ruh, W. (Jul 2004). *Enterprise Integration: The Essential Guide to Integration Solutions*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
- Gorton, I. & Liu, A. (2005). *An Architect's Guide To Enterprise Application Integration With J2EE and .NET*. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005). ICSE '05. ACM, New York, NY, 726-727. DOI= <http://doi.acm.org/10.1145/1062455.1062638>
- Grable, R., Jernigan, J., Pogue, C., & Dale, D. (March 1999). Metrics for small projects: Experiences at the SED. *Software, IEEE*, 16, 2, pp.21 - 29
- Guliani, G. & Woods, D. (Jul 2005). *Open Source for the Enterprise*. Sebastopol, CA : O'Reilly Media, Inc.
- Hart, Robert. (2001). Open source and cooptation for in-house software development. Retrieved February 20, 2007 from <http://www.interweft.com.au/papers/cooptation.html>
- Hewlett-Packard Company. (2001). *Web Services Concepts - A Technical Overview*. Retrieved October 30, 2007 from http://72.14.253.104/search?q=cache:8FCnCoFKQMqJ:se2c.uni.lu/tiki/se2c-bib_download.php%3Fid%3D1409+Together+with+XML,+Web+service+protocol+standards+provide+an+industry-wide+means+to+realize+interoperability+and+platform-neutral+communications&hl=en&ct=clnk&cd=5

- Hohpe, G. and Woolf, B. (Oct 2003) *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
- Intel Inc. (n.d.). *Intel® Core™ Duo Processors*. Retrieved May 1, 2007 from <http://www.intel.com/products/processor/coreduo/>
- Jacoby, G.A. (Jan 2005). Intranet portal model and metrics: a strategic management perspective. *IT Professional*. 7, 1, pp.37-44.
- JA-SIG Central Authentication Service. (n.d.). *Central Authentication Service: Single Sign-on for the Web*. Retrieved February 27, 2007 from <http://www.ja-sig.org/products/cas/>
- JBoss, Inc. (n.d.). *Hibernate Homepage*. Retrieved April 27, 2007 from <http://www.hibernate.org/>
- Jones, C. (1997). *Software Quality*. New York, New York: International Thomson Computer Press, 1997.
- Juric, M. B., Kezmah, B., Hericko, M., Rozman, I., & Vezocnik, I. (May, 2004). *Java RMI, RMI tunneling and Web services comparison and performance analysis*. *SIGPLAN Not.* 39, 5, p.58-65.
- Kalmanovitch, Michael. (n.d.) *AA Newsletter*. Retrieved February 27, 2007 from http://earthsgeneralstore.com/activist_agenda/current/index.htm
- Kauffman, R. and Li, X. (2004), "Payoff Externalities, Informational Cascades and Managerial Incentives: A Theoretical Framework for IT Adoption Herding," MIS Research Center working paper, Carlson School of Management, University of Minnesota.

Kerner, Sean Michael. (Oct 2006). Microsoft Opens PHP Door. *Internet.com Developer*.

Retrieved February 20, 2007 from <http://www.internetnews.com/dev-news/article.php/3641101>

Klaene, Michael. (June, 2004). Understanding the Java Portlet Specification.

Developer.com. Retrieved March 6, 2007 from

<http://www.developer.com/java/web/article.php/3366111>

Krafzig, D., Banke, K., & Slama, D. (Nov 2004). *Enterprise SOA: Service-Oriented Architecture Best Practices*. Hagerstown, Maryland: Pearson Education, Inc.

Kumar, Akshinthala K. (Jul 2005). Data Integration Strategy: EAI or ETL? *DM Direct Newsletter*. Retrieved March 2, 2007 from

http://www.dmreview.com/editorial/dmreview/print_action.cfm?articleId=103296

5

Lahav, Leo. (Nov 2002). Hobbes Framework—an adaptable solution to web-driven applications. *Computer Standards & Interfaces*, 27, 3, pp.271-274.

LaMonica, Martin. (Feb 2005). IBM backs open-source Web software. *CNET News.com*.

Retrieved February 20, 2007 from http://news.com.com/2102-7344_3-5589559.html?tag=st.util.print

Lamonic, Martin. (Jul 2004). IBM plans bus technology ride. *CNET News.com*, Retrieved May 18, 2006, from

http://news.com.com/IBM+plans+bus+technology+ride/2100-1001_3-5262604.html

- Land, R. Crnkovic, I. (Oct 2004).). Proceedings from Fourth Conference on Software Engineering Research and Practice in Sweden: *Existing Approaches to Software Integration – and a Challenge for the Future*. Linköping, Sweden
- Li, X. and Johnson, J. D. (2002). Evaluate IT Investment Opportunities Using Real Options Theory. *Information Resources Management Journal*, 15, 3, p.32-47.
- Linthicum, David S. (Aug 2003). *Next Generation Application Integration: From Simple Information to Web Services*. Boston, MA: Addison-Wesley Longman Publishing Co
- Loke, S. W., & Ling, S. (2000). Proceedings from ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000): *Mobile Agent Itineraries and Workflow Nets for Analysis and Enactment of Distributed Business Processes*. Wollongong, Australia. pp.459-66
- Microsoft Corp. (n.d.). *Prescriptive Architecture: Presentation Integration*. Retrieved May 3, 2007 from [http://msdn2.microsoft.com/en-us/library/ms978588\(d=printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms978588(d=printer).aspx)
- Miller, C. (April, 2003). *End User Programming for Web Users*. In (H. Liebermann, F. Paternò, A. Repenning, V. Wulf): Proceedings of the CHI'03 Workshop on End-User Development. Fort Lauderdale
- Noffsinger, W. B., Niedbalski, R., Blanks, M., & Emmart, N. (Dec 1998). Legacy Object Modeling Speeds Software Integration. *Communications of the ACM* , 41, 12, p.80-89.
- O'Reilly, Tim. (Sep 2005). *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*. Retrieved December 2, 2007 from

<http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1>

Oracle Corporation. (May, 2002). *How Do I use Jakarta Struts with Oracle9i*

JDeveloper? Retrieved December 17, 2007 from

<http://www.oracle.com/technology/products/jdev/howtos/jsp/StrutsHowTo.html>

PHP/Java Bridge Project Team. (n.d.a). *php/Java Bridge Homepage*. Retrieved February

27, 2007 from <http://php-java-bridge.sourceforge.net/pjb/contact.php>

PHP/Java Bridge Project Team. (n.d.b). *PHP/Java Bridge API Documentation - Class*

PhpJavaServlet. Retrieved June 1, 2007 from [http://php-java-](http://php-java-bridge.sourceforge.net/pjb/server/documentation/API/php/java/servlet/PhpJavaServlet.html)

[bridge.sourceforge.net/pjb/server/documentation/API/php/java/servlet/PhpJavaServlet.html](http://php-java-bridge.sourceforge.net/pjb/server/documentation/API/php/java/servlet/PhpJavaServlet.html)

PHP Group, The. (n.d.a). *PHP Manual: strtotime*. Retrieved June 7, 2007 from

<http://ca.php.net/strtotime>

PHP Group, The. (n.d.b). *PHP Manual: strftime*. Retrieved June 7, 2007 from

<http://ca.php.net/strftime>

Reason, P., & Bradbury, H. (2001). Inquiry and Participation in Search of a World

Worthy of Human Aspiration. In P. Reason & H. Bradbury (Eds.), *Handbook of*

Action Research: Participative Inquiry and Practice. London: Sage Publications.

Reyes, A., Espino, J., Mohan, V., & Nadkar, M. (Nov 2003) Proceedings of the 27th

Annual International Conference on Computer Software and Applications

(COMPSAC). *Ad Hoc Software Interfacing: Enterprise Application Integration*

(EAI) when Middleware is Overkill., p. 570

- Shields, B., Owen, M. (2003). Proceedings from International Conference on Internet Computing. *An Agent Based Approach to Enterprise Application Integration.* , Las Vegas, Nevada, USA. CSREA Press. Volume 1, pp. 361-367.
- Singh, Kshipra. (Sep 2006). OS CMS Award Finalists Announced! *NewsForge: The Online Newspaper for Linux and Open Source*. Retrieved February 20, 2007 from <http://newsvac.newsforge.com/newsvac/06/09/12/205222.shtml>
- Stahl, F. & Mass, W. (March 2003). *Empirical Aspects on Content Management*. Paper presented at the Third Open Source Content Management Conference, Cambridge, Massachusetts (Harvard Law School).
- Sun Microsystems. (n.d.a). *JavaServer Pages Technology Homepage*. Retrieved April 27, 2007 from <http://java.sun.com/products/jsp/>
- Sun Microsystems. (n.d.b). *Java SE at a Glance*. Retrieved April 27, 2007 from <http://java.sun.com/javase/index.jsp>
- Sun Microsystems. (n.d.c). *Java API Documentation: java.util.List..* Retrieved June 8, 2007 from <http://java.sun.com/j2se/1.4.2/docs/api/java/util/List.html>
- Suraski, Zeev. (Nov 2002). The Object-Oriented Evolution of PHP. *DevX.com*. Retrieved May 3, 2007 from <http://www.devx.com/webdev/Article/10007>
- Susman, G. (1983). Action research: a sociotechnical systems perspective. In Morgan, G. (Ed.), *Beyond Methods: Strategies for Social Research*. Newbury Park, CA: Sage Publications. pp. 95 – 113.
- University of Washington. (n.d.). *Using a php.ini file*. Retrieved October 29, 2007 from <http://www.washington.edu/computing/web/publishing/php-ini.html>

- Vaughan-Nichols, Steven. (Jan 2002). Fat protocols slow Web services. *ZDNet: Where Technology Meets Business*. Retrieved March 1, 2007 from <http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2836041,00.html>
- Walther, S. (March 2005). Introducing the ASP.NET 2.0 Web Parts Framework. *Visual Studio 2005 Technical Articles*. Retrieved March 6, 2007 from [http://msdn2.microsoft.com/en-us/library/ms379628\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/ms379628(VS.80).aspx)
- Wayner, Peter. (Jul 2005). Enterprise Monitoring Comes to PHP. *InfoWorld*. 27, 27, pp.22-24.
- Wright, Graham. (May 2002). *Extreme Programming In A Hostile Environment* in Proceedings of 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002), Sardinia, Italy, May 26-30, 2002, pp. 48-51.
- Yakimovich, D., Travassos, G.H., & Basili, V.R. (1999). Proceedings from IEEE Computer Society 24th Annual Software Engineering Workshop: *A classification of software components incompatibilities for COTS integration*. GreenBelt, Maryland, USA.
- Grid Today. (Aug 2005). *Yankee Group Issues Results of Enterprise SOA Survey*. Retrieved December 11, 2007 from <http://news.taborcommunications.com/msgget.jsp?mid=462564&xsl=story.xsl>
- Ye, Wangming. (Jan 2005). *Web services programming tips and tricks: Improve the interoperability between J2EE and .NET, Part 2*. Retrieved October 30, 2007 from <http://www.ibm.com/developerworks/webservices/library/ws-tip-j2eenet2.html>

Zuse, H., Drabe, K. (n.d.). *Zuse / Drabe - Measure-Information-System (ZD-MIS)*

Homepage. Retrieved March 13, 2007 from <http://www.horst-zuse.homepage.t-online.de/zdmis.html>

APPENDIX 1

tinycalendar.module – snippet from tinycalendar_block function

```
...
01   if ($op == 'view') {
02   $block_content = "";
03   $JEE_system_get_tiny_calendar_url_and_querystring =
JEE_SYSTEM_GET_TINY_CALENDAR;
04   $request = new HttpRequest($JEE_system_get_tiny_calendar_url_and_querystring,
HttpRequest::METH_GET);
05   try {
06     $request->send();
07     if ($request->getResponseCode() == 200) {
08       $block_content .= strip_markup_from_JEE_response($request-
>getResponseBody());
09       $link_to_current_page = curPageURL();
10       ...
11       //change the "next and previous month" links sent by the JEE system so that
they point to
12       //drupal instead of back to the JEE system
13       $block_content =
14         str_replace('events/get.do?method=loadTinyCalendar&',
15                   $link_to_current_page, $block_content);
16       //replace .do string, and tack on the current date
17       $block_content =
18         str_replace('events/get.do?method=getOneDayOfEvents&currentDate=',
19                   "activistAgenda/viewEventsForDay?" . "currentDate=" .
$_GET['currentDate'] . 20
                   "&dateOfEvents=",
$block_content);
21     }
22   } catch (HttpException $ex) {
23     echo $ex;
24   }
...

```

APPENDIX 2

style.css - snippet

```
1    #header {  
2        background:url(images/header.png) repeat-x;  
3        height:100px;  
4        margin:0;  
5        padding:0;  
6    }
```

APPENDIX 3

web.xml (AA Subsystem) - snippet

```
<servlet-mapping>  
    <servlet-name>action</servlet-name>  
    <url-pattern>*.do</url-pattern>  
</servlet-mapping>
```

APPENDIX 4

activistagenda.module – snippet from activistagenda_eventAdd function

```
01  function activistagenda_eventAdd() {
02  global $user;
03  $request = new HttpRequest(JEE_SYSTEM_EVENTS_CREATOR_ADD,
HttpRequest :: METH_POST);
04
05
        $HTTP_RAW_POST_DATA=isset($HTTP_RAW_POST_DATA)?$HTTP_RA
W_POST_DATA:file_get_contents("php:
06                                //input");
07
08  $request->setRawPostData($HTTP_RAW_POST_DATA);
09  $content = get_secure_page_from_JEE_system($request,
10
        JEE_SYSTEM_EVENT_FORM_ADD_SECURITY_CHECK, $user->pass);
11  $content = strip_markup_from_JEE_response($content);
12
13  $content = str_replace('src=""', 'src="" . JEE_SYSTEM_URL . "/"', $content);
14  $content = str_replace('/activistAgenda/events/creator/add.do?method=add',
15                        '/greenEdmonton/activistAgenda/events/creator/add',
$content);
16
17  $content = str_replace('initRTE("editor/images/"',
18                        '"activistAgenda/editor/"', '"')', 'initRTE("' . JEE_SYSTEM_URL .
'/editor/images/"',
19                        "' . JEE_SYSTEM_URL . '/editor/"', '"')', $content);
20  return $content;
```

APPENDIX 5

activistagenda.module – snippet from event_search_form_submit function

```
01  function event_search_form_submit($form_id, $form_values) {
02      require_once(JAVA_BRIDGE_URL . "/java/Java.inc");
03      $session = java_session();
04      $eventManager = new
Java("ca.activistAgenda.managers.EventManager");
05      $beginRangeDatePosted = null;
...
06      if (strtotime($form_values['datePosted']['beginRangeDatePosted']) !=
false)
07      {
08          $beginRangeDatePosted
09              strtotime($form_values['datePosted']['beginRangeDatePosted']);
10      }
...
11      $searchResults = $eventManager->search( "",
12          $form_values['title'],
13          $form_values['organization'],
14          $form_values['cost'],
15          $form_values['locationName'],
16          $form_values['locationAddress'],
17          $form_values['contact'],
18          $form_values['contactEmail'],
19          $form_values['contactPhone'],
20          $form_values['url'],
21          $form_values['description'],
22          $beginRangeDatePosted,
23          $endRangeDatePosted,
24          $beginRangeEventDate,
25          $endRangeEventDate,
26          false,
27          false);
28
29      $session->put("searchResults", $searchResults);
30      return 'activistAgenda/events/search';
31  }
```

APPENDIX 6

activistagenda.module – snippet from activistagenda_eventSearch function

```
1: function activistagenda_eventSearch(){
2:   require_once(JAVA_BRIDGE_URL . "/java/Java.inc");
3:   $session = java_session();
4:
5:   $searchResults = $session ->get("searchResults");
6:   $iterator = $searchResults->iterator();
7:
8:   while($iterator->hasNext())
9:   {
10:    $event = $iterator->next();
11:    $output .= '<table border="1" cellspacing="0" cellpadding="0">';
12:
13:    $output .= "<tr>";
14:    $output .= "<td><b>Title:</b></td>";
15:    $output .= "<td><b>Organization:</b></td>";
16:    $output .= "<td><b>Location Name:</b></td>";
17:    $output .= "<td><b>Location Address:</b></td>";
18:    $output .= "<td><b>Contact Email:</b></td>";
19:    $output .= "<td><b>Contact Phone:</b></td>";
20:    $output .= "<td><b>URL:</b></td>";
21:    $output .= "<td><b>See More</b></td>";
22:    $output .= "</tr>";
23:
24:    $output .= "<tr>";
25:    $output .= "<td>" . $event->getTitle() . "</td>";
26:    $output .= "<td>" . $event->getOrganization() . "</td>";
27:    $output .= "<td>" . $event->getLocationName() . "</td>";
28:    $output .= "<td>" . $event->getLocationAddress() . "</td>";
29:    $output .= "<td>" . $event->getContactEmail() . "</td>";
30:    $output .= "<td>" . $event->getContactPhone() . "</td>";
31:    $output .= "<td>" . $event->getUrl() . "</td>";
32:    $output .= "<td><a href = \"" . url("activistAgenda/events/load") .
33:               "?eventId=" . java_values($event->getEventId()) .
34:               "\">View Event Details</a></td>";
35:    $output .= "</tr>";
36:
37:    $output .= "</table>";
38:  }
39:
40:  return $output;
41:}
```

APPENDIX 7

activisteagenda.module file - snippet from activistagenda_eventDetails method

```
01 $output .= "<tr>";
02 $output .= "<td><b>Date Posted:</b></td>";
03 $output .= "<td>" .
04 strftime("%a, %b %d %Y", $event->getDatePosted()->getTime()) .
05 $output .= "</td></tr>";
```

APPENDIX 8

activistagenda.module - snippet from activistagenda_eventDetails method

```
01    $parser = new Java("ca.activistAgenda.utilities.Parser");
...
    $output .= "<tr>";
02    $output .= "<td><b>Date Posted:</b></td>";
03    $output .= "<td>" .
04    $parser->convertToDisplayDate($event->getDatePosted()) .
05    $output .= "</td></tr>";
```

APPENDIX 9

activistagenda.module – snippet from activistagenda_eventsview_by_month function

```
...
01  $wsdl = J2EE_SYSTEM_GET_ONE_MONTH_OF_EVENTS_WSDL;
02  $client = new SoapClient($wsdl);
03
04  //Must pass a date_time with this format: 2001-10-26T21:32:52
05  $datetime_of_events = date("Y-m-d\TH:i:s");
06
07  if ($_GET['dateOfEvents'] != ""){
08      $datetime_of_events = date("Y-m-d\TH:i:s",
strtotime($_GET['dateOfEvents']));
09  }
10  $events = $client->getEventsByMonth(new SoapVar($datetime_of_events,
XSD_DATETIME));
11
12  $output .= "<h2>" . date("F\, Y", strtotime($datetime_of_events)) . "</h2><br>";
13
14  foreach ($events as $event)
15  {
16      $output .= '<table border="1" cellspacing="0" cellpadding="0">';
17
18      $output .= "<tr>";
19      $output .= "<td><b>Title:</b></td>";
20      $output .= "<td>" . $event->title . "</td>";
21      $output .= "</tr>";
...
22      foreach ($event->eventDateTimes as $dateTime)
23      {
24          $output .= "<tr><td>";
25          $output .= date("l, F j, Y", strtotime($dateTime->eventDate));
26          $output .= ". From " . date("g:i a", strtotime($dateTime-
>startTime));
27          $output .= " to " . date("g:i a", strtotime($dateTime->endTime));
28          $output .= "</tr></td>";
29      }
...
30      $output .= "</table>";
31  }
32  return $output;
...
```

APPENDIX 10

EventManager.wsdl - snippet

```
01 <complexType name="Event">
02   <sequence>
03     <element name="accepted" nillable="true" type="xsd:boolean"/>
04     <element name="contact" nillable="true" type="xsd:string"/>
05     <element name="contactEmail" nillable="true" type="xsd:string"/>
06     <element name="contactPhone" nillable="true" type="xsd:string"/>
07     <element name="cost" nillable="true" type="xsd:string"/>
08     <element name="datePosted" nillable="true" type="xsd:dateTime"/>
09     <element name="description" nillable="true" type="xsd:string"/>
10     <element name="eventDateTimes" nillable="true"
11       type="impl:ArrayOf_tns1_EventDateTime"/>
12     <element name="eventId" nillable="true" type="xsd:int"/>
13     <element name="locationAddress" nillable="true" type="xsd:string"/>
14     <element name="locationName" nillable="true" type="xsd:string"/>
15     <element name="organization" nillable="true" type="xsd:string"/>
16     <element name="reviewed" nillable="true" type="xsd:boolean"/>
17     <element name="title" nillable="true" type="xsd:string"/>
18     <element name="url" nillable="true" type="xsd:string"/>
19     <element name="user" nillable="true" type="tns1:User"/>
20   </sequence>
21 </complexType>
```

APPENDIX 11

EventManagerSoapBindingImpl.java - snippet

```
01  searchResults = events.getEventsByMonth(month.getTime());
    ...
02  Iterator iterator = searchResults.iterator();
03  Event [] eventArray = new Event[searchResults.size()];
04  int c = 0;
05  while(iterator.hasNext())
06  {
07      eventArray[c++] = (Event) iterator.next();
08  }
09  return eventArray;
```

APPENDIX 12

AA User Manual

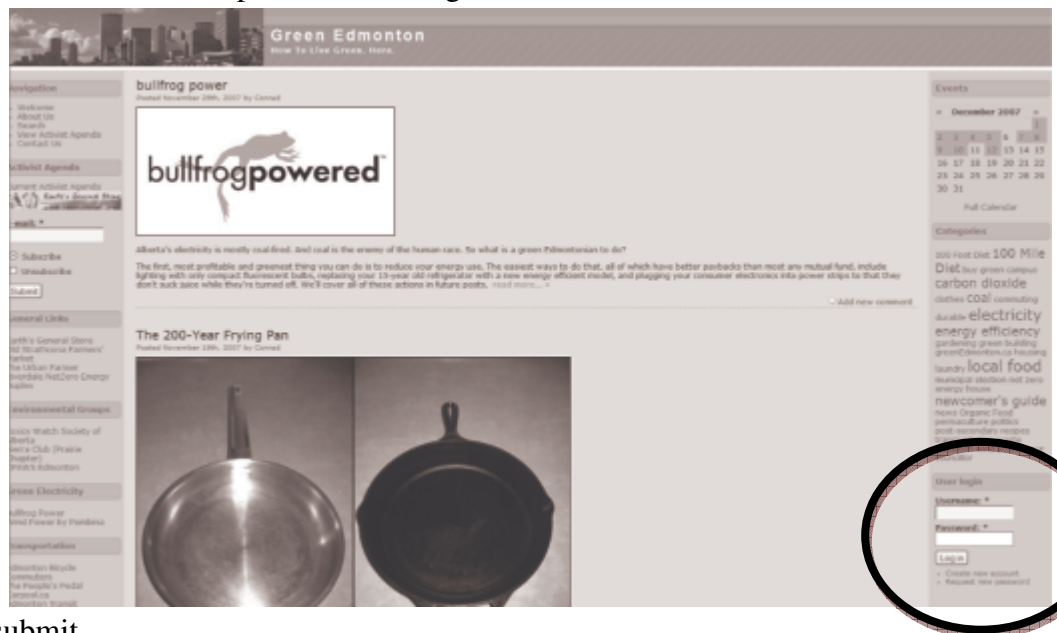
The AA is a weekly newsletter administered by Michael Kalmanovitch of Earths General Store and the Ecology Systems Information Society (ESIS). It contains events and information pieces about the activist community in Edmonton.

As an Event and Item creator, you have the ability to login to the AA system and enter Events and Items. Once entered, they will be reviewed and possibly edited by the AA administrator (currently Michael Kalmanovitch), and, if approved, revealed to the Web surfing public.

Log In

To log into the system:

1. surf to greenEdmonton.ca
2. enter your username and password in the green circled text boxes:



3. Click submit

Event and Item Creator Section

Events

An event occurs on a given date. It may have multiple dates as well as start and end times.

Add an Event


To add an event:

1. Click “AA”:

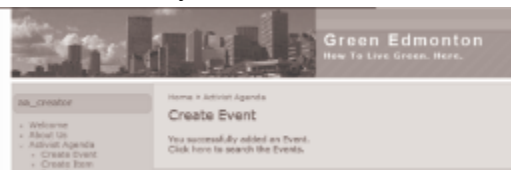


2. Click “Create Event”:



- Fill in the event fields. **Required fields include title, at least one date, organization, and description.** If the event is recurring, you may add up to 12 dates, each with an optional start time and end time. The description field is entered using a rich text editor in which words can be bolded, italicized, and otherwise formatted. To add a date, click "Choose date" to activate the date chooser, and to add a start or end time, click the  icon to activate the time chooser.

- Click submit
- If successful, you will see a screen like this:



- Your event will be reviewed by the AA administrator, and if approved, appear shortly in the AA newsletter:

Items

Items are information pieces that are generally not associated with dates and times. Examples of item types are "Volunteer Opportunity", "Action", and "Information".

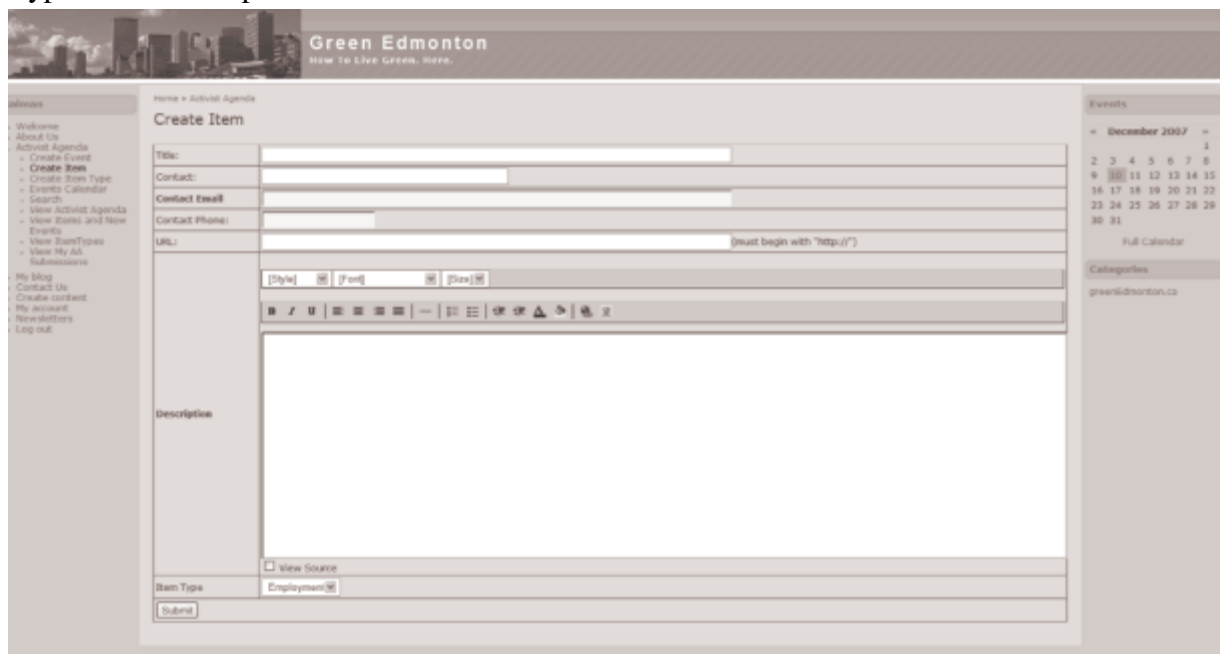
Add an Item

To add an item:

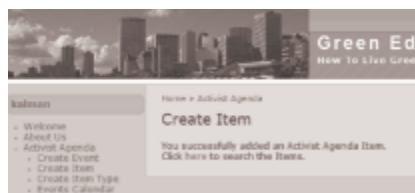
1. Log in (see above)
2. Click “AA”:



3. Fill out the item data fields. Title and description are mandatory. Choose the Item Type from the drop down list.



4. Click Submit:

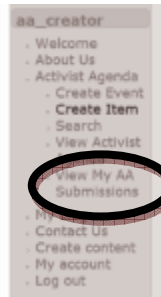


5. Your item will be reviewed by the AA administrator, and if approved, appear shortly in the AA newsletter:

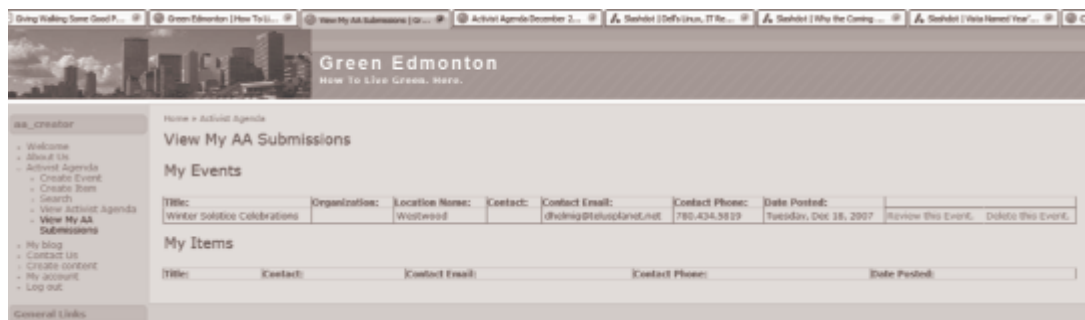
View Your Items and Events

At any time, you may view the Items and Events that you have submitted to the system:

1. Click “View My AA Submissions”:



2. You may edit or delete the event(s)/item(s) if they have not yet been approved by the AA Administrator:

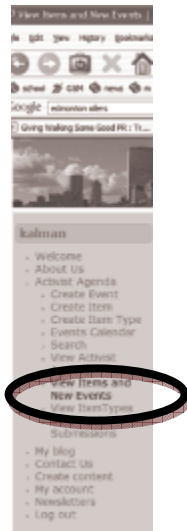


Administrator Section

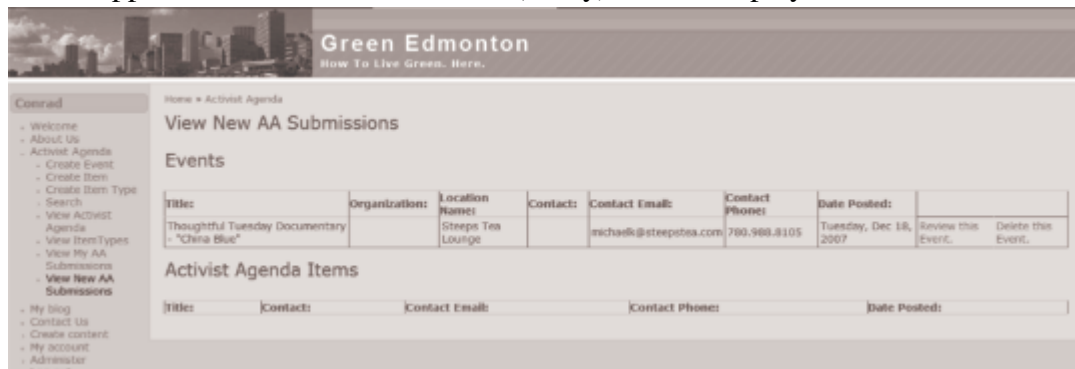
The administrator is the arbiter of good taste and appropriateness for the AA. The administrator approves and edits Items and Events.

Edit/Approve Events

1. Click "View Items and New Events":



2. The unapproved events all of the items (if any) will be displayed:



3. Click on "Review this Event" (see picture above)

- You now have the ability to edit any part of the submitted event. Pay special attention to the "reviewed" and "accepted" checkboxes. These must both be checked in order for the event to be displayed to the public.

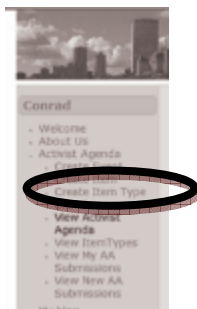
- Once any editing has been completed, and if the reviewed and accepted checkboxes checked if appropriate, click submit.

Create Item Type

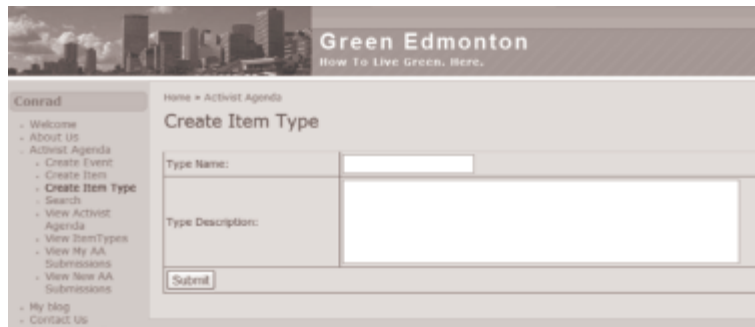
Examples of Item Types are "Volunteering" and "Action".

To create an item type:

- Click on "Create Item Type":



2. Enter the Type Name and Type Description. Both fields are mandatory:

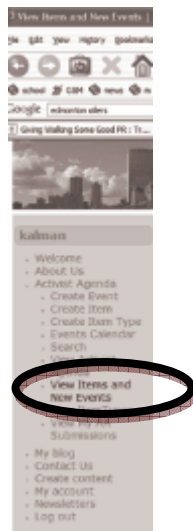


The screenshot shows the 'Create Item Type' form on the Green Edmonton website. The form has two main input fields: 'Type Name' and 'Type Description'. The 'Type Name' field is a single-line text box, and the 'Type Description' field is a larger multi-line text box. A 'Submit' button is located at the bottom of the form. The website header includes the logo 'Green Edmonton' and the tagline 'How To Live Green. Here.'. A sidebar menu on the left lists various navigation options, including 'Create Item Type'.

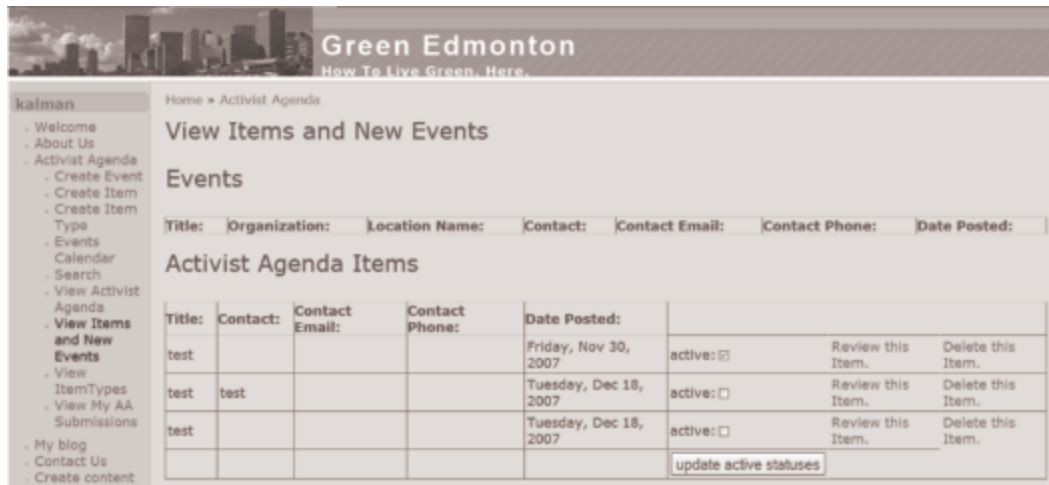
3. Click Submit

Edit/Approve Item:

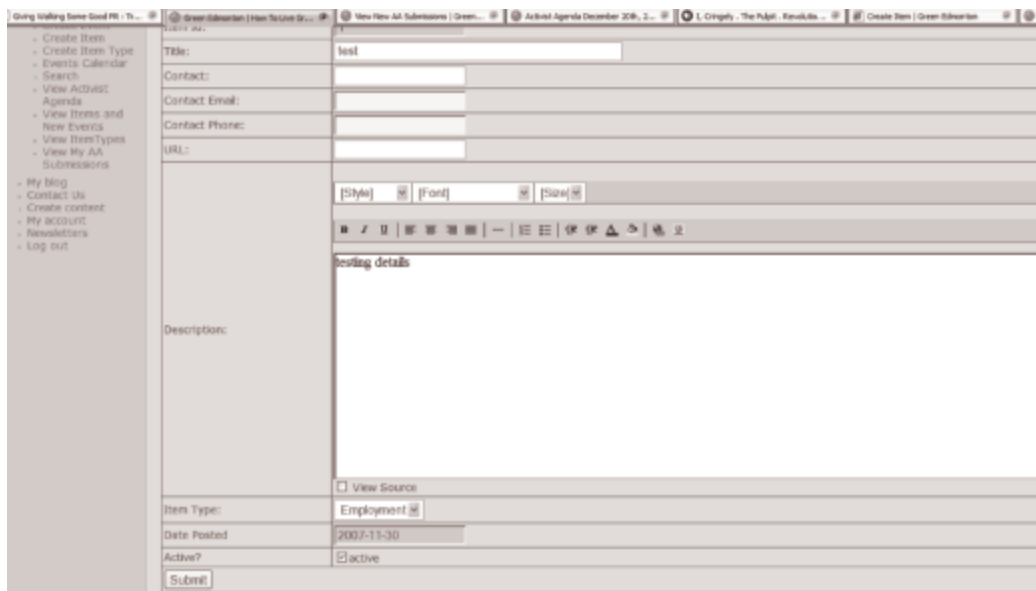
1. Click “View Items and New Events”:



- Click “Review this Item” to review it **OR** select/unselect checkboxes in activate and inactivate items and then click on the “update active statuses” button:



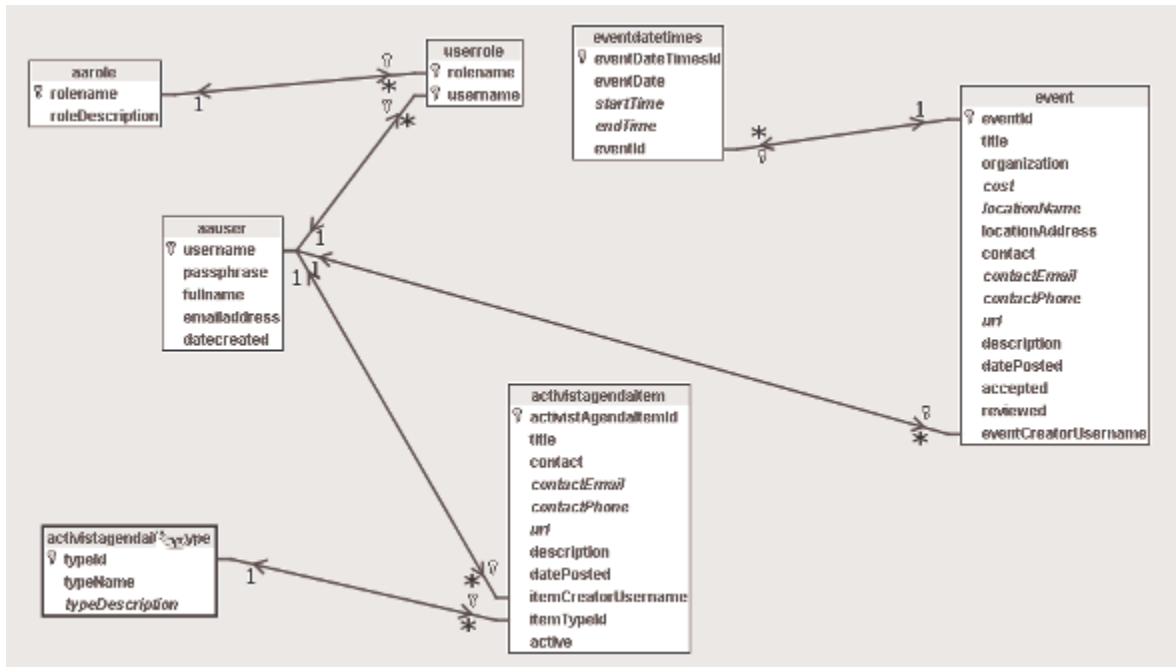
- If you chose to review and item, edit it and check the “active” box at the bottom in order to activate/inactivate it:



- Click submit.

APPENDIX 13

Table of entities: AA Subsystem



APPENDIX 14

Struts Application Framework Architecture (Oracle, 2002)

