Association for Information Systems
AIS Electronic Library (AISeL)

SAIS 2004 Proceedings   Southern (SAIS)

3-1-2004

Building an Understanding of the Service-Oriented Architecture: One University's Experience with Implementing Microsoft's .NET Platform

Meg C. Murray
mcmurray@kennesaw.edu

Follow this and additional works at: http://aisel.aisnet.org/sais2004

BUILDING AN UNDERSTANDING OF
THE SERVICE ORIENTED ARCHITECTURE:
ONE UNIVERSITY'S EXPERIENCE
WITH IMPLEMENTING
MICROSOFT'S .NET PLATFORM

Meg C. Murray
Kennesaw State University
mcmurray@kennesaw.edu

Abstract

A paradigm shift is on the horizon for the way computer applications are designed, developed and deployed. The focus is shifting away from the development structure of 'one program equals one application' to one application made up of a collection of small functional components referred to as services. While the technology is still evolving, it is imperative that information systems programs become proactive in integrating the concepts of the Service Oriented Architecture into their curriculum and providing exposure to its associated tools in the classroom. However, migrating to these new technologies is not trivial. This paper provides an overview of one university's experience of introducing this new paradigm into the college classroom though utilizing Microsoft's .NET environment.

Keywords: Service oriented architecture, Web services, information systems curriculum

Introduction

A paradigm shift is on the horizon for the way computer applications are designed, developed and deployed. The movement is away from the all-compassing large software application to a distributed model where the application is actually made up of a collection of small functional components. Terms surrounding this model often incorporate the word 'service' such as Web services and Service Oriented Architecture. The next generation of applications will be designed around a master template that incorporates functional components as needed and these components may be located anywhere accessible through the Internet.

Envisioning software as a collection of services requires a major change in one's understanding of how software is designed, developed, deployed and utilized. It also requires a new way of thinking for information systems professionals who must manage and implement the new service oriented technology solutions. Consequently, this transformation has many implications for college level and graduate level information systems programs. IS programs need to begin to review what they currently teach and make an assessment of how that can be aligned to this new paradigm. The focus is shifting away from the development structure of 'one program equals one application' to one application made up of a collection of software modules. Students will not only be challenged to understand isolated program development, they will be required to understand program interoperability. This means that Information System programs must actively move to incorporate the theoretical concepts associated with the Service Oriented Architecture into their curriculum and exposure to the associated tools in the classroom. Students armed with this foundational knowledge will be in a better position to become full and active participants in the discipline. For example, some of the most often cited desired skills for technology professionals is knowledge of Sun's J2EE and Microsoft's .NET environments (Swinton, 2003). This paper provides an overview of one university's experience of introducing this new paradigm into the college classroom though utilizing Microsoft's .NET environment.

Overview of Available Platforms

Several major players in the technology marketplace are embracing the new vision. Sun Microsystems, IBM and Microsoft have all introduced frameworks to support a component-based, Service Oriented Architecture (SOA). These platforms provide support for software development, deployment, execution and management that facilitate interoperability across servers, development languages, applications and devices. Sun's Java2 Enterprise Edition (J2EE) and now their Open Net Environment (Sun ONE), IBM's Web Sphere built on J2EE and Microsoft's .Net provide capabilities to achieve this goal but they approach this in different ways. The primary focus of Sun ONE and IBM's Web Sphere is the definition of an underlying architecture through specifications of a standard platform for hosting and delivering J2EE applications built on Java technologies. Microsoft has embraced the SOA by the introduction of their .NET initiative. Microsoft describes .NET as "an open language platform for enterprise and Web development" (Microsoft, 2002). While the frameworks offer similar services, the cliché commonly used to describe them is that '.NET is Windows-centric and language neutral while J2EE is java-centric and platform neutral.'

Each of these platforms offers viable alternatives as they represent very complex and rich environments. Further, both Microsoft and IBM have introduced very attractive programs for higher education. Microsoft offers a program known as the Microsoft Development Network Academic Alliance (MSDNAA) that provides access to all of their software including operating systems, but excluding Microsoft Office, for a very low annual fee. IBM offers a similar program, entitled the IBM Scholars Program, that offers faculty and students free access to their Web Sphere software including their development environment. While academic endeavors should not be vendor-centric, it is not trivial to move to a new development environment, and thus, selecting to focus on environment has its advantages. Consequently, this paper focuses on Microsoft's .NET environment.

Microsoft's .NET is not a product, but as Microsoft states, ".NET is a series of Microsoft technologies interconnecting the world of information, people and devices" (Microsoft, 2002). .NET is a collection of resources that includes development tools and languages, server software and protocols all designed to support the development of Internet based and service oriented applications. The underlying technical components include Visual Studio.NET, Microsoft's multi-language development tool, C#, a new object oriented programming language, Visual Basic.NET, a major revision of the popular Visual Basic language, a Common Language Runtime (CLR) component supporting the inclusion of source code written in multiple programming languages and a new version of Active Server Pages. Foundational to the .NET platform is the incorporation of the eXtensible Markup Language (XML) and the Simple Object Access Protocol (SOAP) that serve as mediums for the exchange of data between applications accessible over the Internet. The thread that ties all these technologies together is the .NET Framework. The .NET Framework is based on two components, The Common Language Runtime (CLR) and .NET Framework Classes, that work together to provide a standard set of data types and standard set of code to perform common functions. The premise of the Framework is to be a common foundation allowing applications, regardless of their source of programming language origin, to access system services in the same way. For example, the Framework supports Internet browser-accessible applications, Web services applications and even Windows based applications.

Implementation Plan

The Service Oriented Architecture, and in particular, Microsoft's .NET initiative present a new way of doing things. While there is much interest in this platform, until recently, most of the focus has been on supporting industry adoption of these technologies. A trend is emerging on the part of vendors, such as Microsoft and IBM, to also assist colleges and universities as well. However, while introducing the concepts, theories and models of the Service Oriented Architecture can easily be mapped to academic pursuits, implementing tools that are designed for the development organization into the classroom and lab environment represents an entirely different set of issues.

There are many considerations that must be made when moving to any new technology platform and .NET is no exception to this rule. These issues range from hardware requirements to deciding on the approach to be taken when teaching application development within the new environment. A phased approach was undertaken to implement .NET into the IS curriculum at a regional university. The goal was to develop a full semester course targeted at upper level undergraduate information systems majors. The pilot project was designed to be implemented in stages spanning three academic semesters. The process was put into motion in the Spring of 2002 when an internal grant was secured to procure a dedicated server to be used for the project. In addition, the University subscribed to the Microsoft Developer Network Academic Alliance. Through this program, software may be installed in departmental instructional labs and may also be checked out to students and faculty, free of charge, to be used for research or instructional purposes.

In the first semester of the project, an upper-level undergraduate student was engaged in an independent study to initially investigate installation and ease of use issues. The focus was specifically on using Visual Studio.NET to develop simple

programs including a Visual Basic.NET application and an ASP.NET application. The results of this work showed that there were many desirable features to the .NET platform but also highlighted some problem areas. Applications could be developed quickly using the Visual Studio.NET environment. However, issues arose in the installation of the product and in the completeness of the documentation. While good documentation and resources exist related to the programming languages, such as Visual Basic.NET, limited documentation was found related to the Visual Studio.NET interface and some of the documentation related to ASP.NET and the associated technology of Web Services was out-of-date or incomplete. While the documentation is improving, much of it still remains out-of-sync and what does exist often does not provide much depth.

In the second semester of the project, another independent study program was undertaken. This time four students were involved. Further, the Visual Studio.NET development environment was introduced to a group of graduate students studying e-business technologies. Students who elected to do so could choose to use ASP.NET to complete the e-business solution project required in the course. The .NET framework and Visual Studio.NET were then installed in the computer lab and the server was made accessible for hosting student work. Through these venues, more experience in using the environment was gained and outcomes resulted in the development of an installation manual, an assessment on the ease of use of the Visual Studio.NET development software and a collection of sample projects. These resources were utilized in the development of a special topics course offered in the Fall 2003 semester.

Utilizing a phased implementation proved to be very beneficial as this approach provided opportunities to overcome unanticipated problems. Even more importantly, it provided the time needed to build strategies and an infrastructure for dealing with problems yet encountered. Many lessons were learned and the most important of these are shared in the following section of this paper.

Lessons Learned

Installation

The first consideration that must be made when moving to the .NET platform is related to available computer resources. There are two areas of concern. One relates to the .NET framework and the other to the Visual Studio.NET integrated development environment. The primary issue with the .NET framework is that it must be installed on the computer used for code development and also on any client machine that will be executing code built using .NET. The framework is available for the most commonly used Windows operating systems and can be downloaded for free from the Microsoft upgrade site.

The Visual Studio.NET environment is an extremely rich tool including support for development in four languages (Visual Basic.Net, C++.NET, C# and J#). Also included is support for ASP.NET, XML and the development of Web services. In addition, Visual Studio.NET includes other functions such as support for automatic access to databases, mail servers, and other system resources. Visual Studio.NET has 13 main tool windows that can be customized to meet individual user needs. A default scheme is selected when a specific language is chosen for development and it is often best to use this default scheme when first learning the tool. Spending time upfront to build a comfort level with using the different tools is well worth the effort.

Many challenges and issues arose during the installation process. Visual Studio.NET requires a fairly fast processor and uses a fair amount of system resources. For instance, minimum requirements are a 600MHZ processor, 128MB of RAM and a minimum of 4GB hard drive space. A CD or DVD is required to install the software and installation time can extend to well over 2 hours when no problems are encountered. The installation comes on either one DVD or 5 CDs and while it can be distributed over the network, the download time is prohibitive without fast access. Visual Studio.NET is only supported on the Windows 2000, Windows NT and Windows XP operating systems. (It should also be noted that while Visual Studio.NET will run on XP home edition, it can only be utilized to develop Windows based applications such as those written in Visual Basic.NET. In order to utilize Internet programming such as ASP.NET or Web services, the development computer must support the IIS Web server which is not included as a part of XP home edition.) These requirements are a concern when assessing the resources available in the classroom, student labs and on student owned computers. Many students are not running the latest operating systems and even though they can be distributed to students under the MSDN Academic Alliance, an operating system upgrade is a major undertaking. This restriction means that institutions moving to the .NET platform must be prepared to offer substantial student computer lab support and if distribution to students is an option, one must be prepared to spend the first couple of weeks of the course assisting students in the installation process.

Installing Visual Studio.NET in a laboratory environment has its own set of challenges. Visual Studio.NET requires user permissions in order to perform functions such as debugging and in the case of ASP.NET, write permissions must also be granted to the default web server (IIS in this case) directory. Many of these can be circumvented if students are allowed to log in as 'administrator.' However, this is not a practical solution as it means more time must be spent in managing

laboratory computers. Successful implementation on lab based machines means spending time to develop user profiles and testing them to be sure adequate permissions have been granted.

Vocabulary

Along with this new development environment is a new vocabulary. Microsoft relies heavily on the use of 'controls.' Controls refer to resources or functional entities that may be easily incorporated into an application. Many controls are provided within Visual Studio.Net but they can be secured from an outside source as well. They are generally grouped within categories such as User Controls, HTML Controls, Data Controls and Server Controls. User controls are built by users and are often designed to be shared by other application developers. Examples include calendar controls, etc. HTML controls provide commonly recognized HTML elements that can be incorporated into an ASP.NET form. (The ASP.NET form is run on the server side and results in HTML pages that are sent back to a client browser.) Data Controls provide automated support for connecting to database resources. Server Controls is simply a term used for controls that run on the server. Each control provides its own set of properties and behaviors so it is imperative that students quickly grasp the concept. Student with experience developing in earlier versions of Visual Basic will have familiarity with controls, however with the release of Visual Studio.NET Microsoft has greatly expanded their use.

While controls provide automated support for program development, they are limited. Depending on the control, the act of dragging a control to a form causes lines of code to be automatically generated. However, many controls require the developer to add additional lines of code. These lines of code, whether automatically generated or manually entered, are placed into what is called the "Code Behind" page. Therefore, development in this environment begins with opening a blank form, adding controls to the form, and then moving between the form and code behind page to insert any additional lines of code that might be needed. For instance a 'button' control may be placed on a form but the actions to be taken when the button is clicked must be manually entered in the code behind page.

The set of controls representing the greatest challenge are those related to database connectivity. Visual Studio.NET includes data controls that provide wizards that result in automatically generated code for database connections and query building. Database connectivity is always a challenge for students so this support is welcomed. However, the data controls proved to be fairly complex and cumbersome to use. Visual Studio.NET incorporates two approaches to database interaction; one follows the more familiar flow of establishing a connection and then issuing commands to gain access to the data or make changes to the database. The second approach, fully implements what Microsoft terms ADO.NET. The major feature is the separation of the data source from the data manipulation. This incorporates the use of datasets. A datasets is described as a copy of the data maintained in memory and is referenced within Visual Studio.NET with an XML schema. Data in the dataset is acted upon. When all changes have been made, the dataset is then uploaded to the database. In addition, Visual Studio.NET provides a datagrid object which can be mapped to a view of the dataset. The datagrid includes support for the display and editing of the data. While the ideas behind the datagrid are very useful, it is very complex, documented support is not always accurate and consequently, it does not always work as expected. The data controls proved to be one area that still needs to mature.

In order to build applications within any development environment, one must grasp the how and why certain procedures, processes and principles are utilized. This first step - developing an understanding of the principles and building a 'comfort level' with applying them - is a prerequisite to success and a necessary part of any course dedicated to teaching using the .NET platform.

Deployment

An area that required much attention but was not anticipated was the deployment of student projects. The environment is designed so that development and the initial building of applications take place on a developer's personal computer. When a project is completed, it is then deployed to a production system. The process is straightforward when this happens within an environment where parameter settings and security mechanisms can be easily controlled and managed. This is not a characteristic of the academic classroom environment. However, these types of controls are mandatory given the fact that service oriented applications make reference to resources not necessarily contained within the application itself. Accordingly, there is an added layer referred to as configuration management that must take place to insure that when an application is moved from one machine to another, references are not lost. Recognizing this, Microsoft designed it so that each Internet based application, such as an ASP.NET application, would include their own configuration file. Syncing multiple configuration files is not trivial. Many times a student project would run fine on the student's own system but when uploaded to the server would not work without much tweaking of configuration parameters. This was frustrating for students and difficult for faculty who need to review the students' work. The easiest fix to this was to use only one general configuration file and ask students to either rename or delete the configuration files generated for each application they developed. Implementing standards, policies and detailed instructions for uploading projects became a necessity.

Applications built for the Service Oriented Architecture are not isolated entities and insuring the interoperability of the components they contain is now a major part of application development.

Conclusion

The software applications of tomorrow will be interrelated, distributed peer-level software modules and components that work in tandem to achieve specified functional goals. These complex software systems encompass a collection of distributed applications, resources and services connected via a network of computing systems. This software as a service model is based upon a build-via-assembly approach and this type of development will require new ways of thinking about software architectures and software delivery. These new ways of thinking must begin to permeate what and how computing technologies are taught and even though the technology is still young and evolving, the time has come to find ways to incorporate these new platforms into the curriculum and into the classroom.

References:

Douglass, B. P. (2001, January). The evolution of computing. *Software Development Magazine*, 01(1).

Farley, J. (2001, March). Picking a winner: .NET vs. J2EE. *Software Development*, 9(3), 36-50.

IBM. The IBM Scholars Program. (n.d.) Retrieved December 20, 2003, from www-3.ibm.com/software/info/university

Lauer, C. (2001, January). *Introducing Microsoft .NET*. Retrieved December 20, 2003, from www.dotnet101.com/articles/art014_dotnet.asp

Meyer, B. (2000, November). The Significance of 'dot-NET.' *Software Development*, 8(11), 51-60.

Murray, M. (2003). Move to component based architectures: Introducing Microsoft's .NET platform into the college classroom. *Journal of Computing Sciences in Colleges*, 19(3). 301-310.

Microsoft Corporation. (2002, April 04). Defining the basic elements of .NET. Retrieved December 20, 2003 from www.microsoft.com/net/defined/whatis.asp

Microsoft Corporation. (n.d.) MSDN Academic Alliance Program. Retrieved December 20, 2003 from www.msdnaa.net/program

Swinton, A. (2003, January 7). .NET may top list of job skills in demand. Retrieved December 20, 2003 from zdnet.com/com/2100-1104-978922.html
Database, 26(2 & 3), 20-41