SOFTWARE PROJECT COMPARISON: CONVENTIONAL DEVELOPMENT APPROACH VERSUS CLOUD COMPUTING

Wook-Sung Yoo, Ravindra Kollapu
Fairfield University
1073 North Benson Rd, Fairfield, CT 06824/USA
wyoo@fairfield.edu, Ravindra.Kollapu@fairfield.edu

ABSTRACT
The corporate world is moving towards cloud computing rapidly in order to reap the potential economic benefits of reduced costs, infrastructure assets, and resources. This paper examines the advantages of cloud computing by comparing it to more conventional software development techniques. Specifically, this paper presents a side-by-side comparison of the effort and resources required to build a software tool using: (1) a conventional development solution using .NET framework with an SQL Server database and (2) a cloud computing vendor service solution using Force.com. This comparison allows us to understand the concept of using cloud computing and to determine where cloud computing is most beneficial to clients from the perspective of ease of development, lower production costs, and lower maintenance costs.

KEY WORDS
Software Engineering, Cloud Computing, PaaS, SaaS, .NET, Force.com

1. Introduction

In the last few years, cloud computing has expanded steadily as an infrastructure, platform as a service (PaaS), and software as a Service (SaaS). The corporate world rapidly moves computing resources from physical data centers residing at the company to web based remote computers which can be accessed and maintained instantly and on an unlimited basis [1,2]. SaaS offerings such as Amazon's EC2, Salesforce's Force.com, or Google App Engine, provide alternatives to the traditional on-premises model of software application development and delivery [3,4]. The Cloud Computing phenomenon likewise creates exciting challenges and opportunities for the entire educational system and cloud computing has been widely recognized as a critical component in the field of education [4,5]. However, there is little research explored to define a framework and evaluate the benefits regarding cloud investment [6,7]. Software Engineering program at Fairfield University introduced the use of cloud computing through Capstone project course and this paper present the findings from the case study. The Capstone project team implemented an Online Advising Tool to help department chairs advice students and set up study plans for each student. The Online Advising Tool application was developed in two different ways using two different platforms: (1) a traditional software development platform using .NET Framework with Microsoft SQL Server database, and (2) a cloud computing platform using Force.com. To select the platforms, the team examined several different options in traditional approaches including PHP with MySQL, Enterprise Java, and .NET and also examined several options of cloud service including Amazon Web Services, Microsoft Azure Service Platform, Google App Engine, and Force.com. The team couldn’t identify any major different among those options and decided to select .NET for conventional approach and Force.com for cloud option mostly due to the familiarity of the technology to the most members in the team. These parallel development efforts were done in order to identify the specific advantages of each platform. The following sections examine and compare the effort and resources required to develop the Online Advising Tool using both approaches.

2. Online Advising Tool

The Online Advising Tool (OAT) is an online tool developed for department chairs to manage the study plans of students in the School of Engineering at Fairfield University. OAT streamlines the study plan process, reduces time and cost associated with alerting students for registering for classes, and alleviates the manual entry of information performed by the department chairs.

The Online Advising Tool includes the following main features with four user types (admin, chair, student, and system user):

- Academic program management
- Catalog management
- Student account management
- Study plan management

Figure 1, below, is the use case case diagram that describes the functions of each OAT user type.
3. Implementation of OAT

Once the requirement analysis stage of OAT was completed, two groups of Capstone project students developed the system, one using .NET and the other using Force.com. .NET group consists of four Masters students who were familiar with .NET Framework and the Force.com group consisted of only one Masters student who was familiar with Force.com. Force.com was selected as the SaaS platform for the project because it does not expose developers to its infrastructure unlike the other SaaS products. Force.com also allows customized applications centered around a relational database. It is proven to operate at a higher level of abstraction and dramatically improve productivity for developers and return for investment [8]. The kind of projects suitable for the Force.com platform are those that are data centered which require storage of data, user interface comprising of wizards and grids, requires security and role-based access supporting business processes such as email, spreadsheets and workflow management. The requirements for the Online Advising Tool matched well with what Force.com can provide.

3.1 Architecture

A. Conventional Approach using .NET

Model–View–Controller (MVC) is a computer software design pattern that separates the representation (View) of information from business logic (Control) and data layers (Model). MVC has been widely adapted as a standard architecture for Web applications [9,10]. In order to develop the solution using .NET Framework in MVC, the team downloaded the MVC framework and C# code was written in a precise manner to implement the three layers: View, Controller, and Model [11]. Code had to be written to provide services such as logging, transaction processing, and workflow management. The database implementation required building the database and creating views and stored procedures explicitly for the solution to function in SQL server. A high level system diagram is provided in Figure 2.

B. Cloud Computing Approach using Force.com

Force.com Services are divided into database, business logic and user interface. The architecture of Force.com is based on a standard MVC pattern as shown in figure 3 [8]. Since this is an out of the box feature provided, there was no requirement to code or develop the solution to implement MVC. The architecture framework also provides infrastructure with metadata, relational database, and application services including logging, transaction processing, validation, workflow, and email. Once developers build objects defining tables and fields, stored procedures are implicitly built by the Force.com framework.

3.2 Database Essentials

Based on requirement gathering, database was designed and Entity-Relationship diagram (ERD) of OAT is provided in Figure 4.
A. Conventional Approach using .NET

In the conventional approach, the database, tables, fields, views, and stored procedures had to be built to create the database in SQL server. The relationships between tables such as one-to-many / many-to-many / one-to-one relationship were defined as well as master detail and look up relationships. Stored procedures were added to insert, update and delete data from the tables. Referential integrity was maintained in the database through constraints. Custom built SQL Database diagram of OAT is displayed in Figure 5.

B. Cloud Computing Approach using Force.com

Force.com provides a well integrated logical database. The relational model maps to the business entities which are displayed as user interface components. Declarative metadata is the structure of the database; it is available in xml to be configured in another environment. Data is queried through Salesforce Object Query Language (SOQL) and Salesforce Object Search Language (SOSL) [12]. To implement the OAT solution in Force.com, the developer only needed to create objects, define the fields for the objects, and define a Master Detail type field or a look up field type. There are other standard data types that can be defined that translate to the type of control that is displayed on the user interface. Force.com allows you to define tables containing about 500 fields and define field data types such as currency, pick lists, as well as the other standard data types. Figure 6 shows the object defined in OAT and Figure 7 is the database schema for OAT in Force.com.

3.3 User Interfaces

A. Conventional Approach using .NET

User interface of OAT was built using.aspx. Each use case had a separate aspx page where the controls were added, binding controls to data and wiring control events were explicitly implemented. All interactions to control
the presentation layer and the business logic layer had to be coded in the controller class. All validation rules and checks to maintain the integrity of the data were coded in the aspx.cs. Grids were implemented to display the related lists such as study plan courses for a study plan. In order to build a search layout, a new aspx page had to be created with a Grid View bound to a data source. The events for the grid view control had to be wired to respond to any actions performed on the grid. A snapshot of study plan management page using .NET is displayed in Figure 8.

**Figure 8**
Study Plan Management Page in .NET

**B. Cloud Computing Approach using Force.com**

Building the user interface in Force.com was seamless. Once the object model was designed, each field in the object was designated a data type which corresponded to the control type displayed on the screen. Events and binding data to the controls were handled by Force.com. Validation rules, formula fields could also be set within the page. The user interface components such as list views, enhanced list, related lists and detail components allow rendering of data such as in a grid. The user interface implemented Ajax controls that allowed page refreshes to occur unnoticed by the end user. All of these items were built in features of Force.com. During the design of the object node, the page layouts were also set which set the fields that were displayed on the screen. Security settings also were set to in order to display different page layout for different profiles. The User Interfaces in Force.com are typically created using page layouts and Visualforce pages which allow the developers to build custom interfaces. Page layouts are derived from the table layouts and Visualforce pages are mainly driven by meta data which use the definition of fields in the database to provide the right user interface without custom code. For example, creating a Visualforce page with an input field mapped to a date field in the database renders a calendar picker component consistent with the Force.com native user interface. Figure 9 shows the snapshot of the study plan management page created using Force.com.

**Figure 9**
Study Plan Management Page in Force.com

With Force.com, the framework builds predefined search and dialog layouts in which the user can define the elements to be searched easily. Figure 10 shows the snapshot of search course page in Force.com.

**Figure 10**
Search Course Page in Force.com

**3.4 Business Logic**

**A. Conventional Approach using .NET**

The .NET solution required controller classes to be built for each aspx. Each use case translated into a different screen which had to implement a custom class that controlled the aspx page. The controller class communicated with the data access layer through the WCF (Windows Communication Foundation) services. The data access layer was responsible for interacting with the database and retrieving all the data required for the page. The Model which consisted of entity classes was hydrated through the service. The controller classes used the entities to use the data to perform functions required for each user interface. LINQ was used to filter the data retrieved. All CRUD (Create, Read, Update, and Delete) actions had to be explicitly defined. The View
implemented events for controls to respond to the action. Each CRUD action called a method in the controller class which passed the required parameters to the service which in turn called methods that executed stored procedures to insert data into the database. All role-based security had to be defined in the class. Based on the user type, the option to view and edit were restricted in the user interface. Validation rules were coded in the controller class to prevent invalid values from getting saved in the database. All control events were defined to respond to user actions.

B. Cloud Computing Approach using Force.com

Building the OAT solution in Force.com did not require additional Apex classes to support custom behavior or Visualforce pages for a custom user interface. Apex is Force.com’s own programming language, a combination of Java and T/SQL, which can be written using a browser or as plug-in to Eclipse IDE. The business logic was implemented using Lookup and Master Detail data types for the fields for various objects. Data is queried using SOQL and SOSL. Validation rules were set to trigger invalid values, workflow rules were set to approve and decline a study plan change made by the student. All the CRUD operations were implicitly handled by Force.com. Business logic allows you to use apex code to develop classes, objects and interfaces. Data binding allows you to bind the tables to the user interface elements. Validation rules, formulas and field history tracking provide ways to maintain the integrity of the data. Managing users, roles, profiles, public groups are built in features of Force.com.

3.5 Security

A. Conventional Approach using .NET

Authentication and authorization had to be implemented from scratch for the .NET solution. To achieve profile or role based access, the OAT_UserType table had to store the types of users allowed in the system and the OAT User table had to implement a UserType field for each user. During the login procedure into the website, the application determines the type of the user and sets the privileges for each user type (Admin, Chair, or Student). All the security features had to be explicitly handled and coded.

B. Cloud Computing Approach using Force.com

Force.com provides a multilayered structure for data security. Profiles provide object level security to allow users to read, edit, and delete records of each object. Using Field Accessibility, a feature that makes the field available, fields within the profile can be set to Required, Editable, Read only, or Hidden. The settings in the profile are given higher preference than the page layout settings. The OAT tool implements three different user profiles: Admin, Chair and Student. Three different profiles were defined and users were added to each profile. Based on the profile, the page layouts for a study plan can differ between a Student and a Chair. This was handled by using Record Types. Force.com also provides IP authentication and login hours. These security features ensure that only authenticated machines can use the application. Developers only need to define authentication and authorization as these features were implemented implicitly.

3.6 Deployment

A. Conventional Approach using .NET

.NET solution was deployed on IIS 6.0 web server with SQL Server 2008, and .NET Framework 3.5 installed on the web server. The objects, stored procedures and views were created in SQL Server.

B. Cloud Computing Approach using Force.com

Deployment in Force.com is done by moving custom objects and fields, workflow rules, Apex classes and triggers, and other Force.com components from one environment to the production environment. The Force.com Migration Tool, a Java/Ant-based command-line utility, was used to move metadata from a local directory.

3.7 Development Time Comparison

Excluding requirement gathering and analysis phase, the total hours spent on creating .NET solution by group of four developers was approximately 1250 hours in total and the system was delivered after five months’ efforts while total hours spent using Force.com was 120 hours and system was delivered in two months by one developer. However, please note that students involved in .NET solution may not have the skill set at the level of professional developers so that some time-saving advanced features of .NET solution could have been missed, which could have reduced the development time considerably if it was implemented by a professional development team. However, the same can apply to Force.com solution team who implemented the system while learning more about Force.com technology. Based on the results, the advantage of using Force.com over conventional approach in development time was clear.

4. Conclusion

Building the solution on both the platforms gave us a better understanding of the advantages and the disadvantages of the both approaches. Cloud computing solutions using Force.com reduced considerable amount of efforts of customization with its built-in workflow.
Once the object model was implemented properly based on the requirement, the rest of the development such as the user interface, actions, events validations, workflow rules, security were pre-built for the customer. Same application built on conventional development platform, .NET Framework in this case, requires implementing those features from scratch which costs more and at its own risks. Benefits of building solutions using Force.com include (1) building solutions fast and secure, (2) easy Integration with external web services, (3) integrated workflow and reporting features, and (4) low maintenance cost with high uptime. The benefits of using cloud computing are clear especially for student developers who do not have advanced technical skills and have less experience in developing large scaled commercial level software systems. However, Force.com also put limitation including the limited data volume, restrictions on the number of requests in a transaction, the duration of each request, and the size of each response and request, etc. More importantly, several challenges are being raised from the adoption of this computational paradigm including security, privacy, and federation [13]. Many issues such as securing data remain as the hurdle in wide adoption of cloud computing [14]. Potential problems of service interruption, dependence on external server and service, and the need of constant internet access to cloud computing can be another concern to run business critical application on cloud. It is hard to conclude which approach is superior to the other approach with one experiment and more research should be followed but cloud computing provides potential rich benefits and it could meet needs of education and corporate world in certain applications.

Acknowledgement

Special thanks to Capstone Project course students who developed OAT in Force.com and .NET.

References