Cloud Computing for Capstone Software Development Courses

Robert F. Roggio
broggio@unf.edu
Computer and Information Sciences
University of North Florida
Jacksonville, FL

Abstract.

This paper proposes a new approach to teaching a two-course sequence capstone course in software development. Rather than having students use accustomed methodologies, such as the Unified Process, Agile, Lean, or other hybrids as methodologies, this paper proposes making cloud computing a core issue in this sequence. Teams are divided into those providing infrastructure, platform, application support, coupled with a client base that uses the cloud to develop applications.

Keywords: Cloud computing; infrastructure layer, platform layer, application layer, traditional software development capstone courses.

1. BRIEF MOTIVATION FOR A RE-ORIENTATION OF OUR CAPSTONE COURSES

Oftentimes student groups produce impressive applications in our familiar two-course senior design courses or other similarly-named sequences. While many of these applications may not be industrial strength, students often do a very fine job. But the concern this paper raises is in not recognizing the rapid migration toward cloud computing as the framework for many modern application needs.

Our students must be able to respond to state of the art development strategies in an every-changing workforce. Students must be aware that what they are developing must fit into a much larger, encompassing enterprise-level picture where bottom-line economics, time to market, competition, control of data, security, and compliance are huge issues. These topics are rarely, if ever, covered in these courses because instruction and group efforts center on developing applications. It is time to address this shortcoming.

2. THE TRADITIONAL CAPSTONE SEQUENCE

We often start with business modeling that addresses a business vision, risks, business flow and business objects. More often than not, specifications may be captured with use-cases, where there are as many opinions in their formatting and refinements as there are definitions of cloud computing. A development approach central to the project may include an agile approach or one of its variants; perhaps the Unified Process; maybe a bit of 'leanness.' We then fully engage students specifying, developing, implementing, and testing an applications project as part of a capstone experience.

Often, but not always, our capstone sequence may be a single course, a couple of courses with related titles, or a clear sequence, such as Senior Project 1 and 2. There are, in fact, many different models for these courses and conducting these is a complex undertaking (Clear, Goldweber, Young, Leidig, Scott, 2001). Further, the content of these courses vary a

©2011 EDSIG (Education Special Interest Group of the AITP)

great deal (Bryant, 2001) Most capstone sequences feature several presentations at various breakpoints or milestones in the development effort.

3. CLOUD COMPUTING CONCEPTS, AND ABSTRACTION

Cloud Computing Definition

Recently, in combatting the many standard methodology wars among Agile, Lean, Unified Process, others (Kennaley, 2010), cloud computing has entered onto the horizon and is rapidly gaining acceptance in various applications domains. Cloud Computing can be described as essentially an on-demand self-service Internet infrastructure where a company may pay-as-it-goes for support and use only what resources it needs. Access is obtained via a browser, an application or some API. Enterprise strategists look toward cloud computing as a significant cost-saving efficiency measure, and the widespread movement extends all the way to the national level as evidenced by a movement to push cloud-based services across the federal IT landscape (Crews, 2011)**.**

According to http://blog.sonian.com/bid/31553 even from recognized authorities in the discipline, there is little common agreement as to exactly what constitutes the cloud. In truth, cloud computing is reasonably new and is evolving so very quickly people have difficulty in clearly defining what 'the cloud' really is.

Cloud Computing and Abstraction

Cloud computing is the encapsulation of many concepts, and the underlying features of the cloud need to be clearly understood by those in the capstone courses so as to provide a framework for developing a cloud application. Fundamental, yet tacit understanding address computation, software, data access, and storage services not requiring end-user knowledge of the physical location and configuration of the system delivering services. Specifically, the user is unaware of the how the services are actually provided. It is this abstraction that is central to appreciating the cloud.

Just as in a standard data structures course where one needs to grasp abstract properties, so too in trying to comprehend cloud concepts, students need an awareness of four important core concepts. These terms and their meanings are not new, yet they underpin understanding of how the cloud works: virtualization, service oriented architectures (SOA), automatic, and utility computing.

Virtualization in our context refers to both the creation and management of a virtual machine. A host control program simulates a powerful computing environment consisting of a number of guest virtual machines. So, while the concept of virtual machines is taught to students in operating system courses it is the understanding of the application of virtual machines that is absolutely essential to student understanding in discussing the cloud. In the capstone course, the application of this concept must be driven home.

A service-oriented architecture system brings together a suite of interoperable software components that can be used for many systems in multiple and diverse application domains. SOAs may cooperate via established protocols and messaging, and via an 'orchestration' process, service-providing objects are gathered and organized into network-accessible units providing services to users building applications (Bell, 2008)**.** These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services. (Bell, 2010)

The term autonomic refers to the self-managing characteristics of some distributed computing resources which are able to adapt to changes as part of a self-managing effort to stem the growing complexity of computing systems tending to set up barriers to additional growth. (Xiaolong & Liu, 2004)

Certainly not a new idea, utility computing has nevertheless quickly become the cornerstone of 'on demand' computing, Software as a Service (SaaS), and cloud computing. The basic principle is simply charging for services provided. This notion of services at a price is fundamental to understanding cloud computing. Services typically provisioned include storage, computations, networking and other desired features. Resources needed are simply 'rented.'

So cloud computing is a natural evolution of the widespread adoption of virtualization, service-oriented architectures, autonomic and utility computing. Details of the desired resources are abstracted away from the users, while traditional expenses of having one's own high levels of expertise and local control over expensive computing resources are no longer needed.

A definition of cloud computing offered by the National Institute of Standards and Technology (NIST) is: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services, that can be rapidly provisioned and released with minimal management effort or service provider interaction." http://www.csrc.nist.gov

4. CLOUD ARCHITECTURE AND PROVIDERS

Cloud computing is normally agreed to be broken up into multiple segments that include a Cloud Infrastructure, Cloud Platforms, and Cloud Applications.

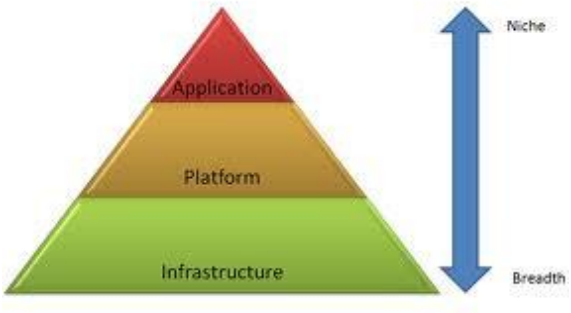

Figure 1 Cloud Architecture (Walczak, 2010)

One can envision cloud computing as a pyramid, where the base of the three-dimensional triangle is referred to as the infrastructure, the middle level may be referred to as the platform, and the upper level may be referred to as the application (Crews, 2011)

The Infrastructure

The Infrastructure, or bottom layer of the Cloud Pyramid is the delivery of computer infrastructure through providing servers, networks, and other hardware appliances delivered as either infrastructure Web Services or 'cloud centers.' Full control of the infrastructure is typically provided as a service at this level.

In down-to-earth terms, the use of this layer involves using what 'comes out of the box,' but not all boxes are equal. In principle, there are many dependencies in using an established infrastructure such as what is and is not commonly provided depending on the cloud provider). And these do vary. These ideas need a bit more explanation. It is this layer of completely virtualized IT configurations that forms the actual foundation of the pyramid. This foundation is absolutely critical to the smooth functioning of the pyramid, as it is this infrastructure upon which cloud platforms and applications are built. Companies providing infrastructure are built to enable cloud platforms and cloud applications. That said, some companies who operate their own infrastructure may provide more features, services and control than others. Often Windows or Linux servers are virtualized so that they can be provisioned dynamically and be on demand for specific needs. Other types of infrastructure components and other services such as storage that may frequently be scalable on-demand cloud storage also exist within this layer. These providers offer a full assortment of network and infrastructure devices for an end-to-end solution. The Cloud Infrastructure is often referred to as 'Infrastructure as a Service' (IaaS). (Crews, 2011)

Infrastructure Providers

There are a number of excellent providers of IaaS. Amazon EC2 (Amazon Elastic Compute Cloud, Eucalyptus (open source cloud platform used for private clouds), IBM, Joyent, Microsoft Azure VM role and others are worthy of investigation. Key features of these include their basis on virtualization of machines and networks, instant provisioning, and much more. Understanding virtualization is important when working with IaaS. As a sample, Amazon Web Services (AWS) provides a suite of elastic infrastructure services that are available on demand that enable requisition of compute power, storage, and more. AWS enables users to launch and manage server instances. (Czarnecki, 2010) Other Amazon infrastructure services include Amazon Simple Storage Service, Amazon SimpleDB, and more. There is a considerable amount of information readily available on the web. A MySQL server can be deployed on Amazon RDS (Relational database Service).

The Platform (Middle) Layer

The middle layer of the Cloud Pyramid provides a computing platform or framework as a service. Control is limited to what the provider provides and the facilities available within it. Clients considering the cloud need to look into available platforms, because each platform provider has limitations. A client cannot rent what is unavailable.

Once a cloud platform is adopted, developers may then build applications or run applications on the provider's infrastructure. While a client may relinquish management and control arising from owning their own IT shop, they are saved from developing, maintaining, staffing and otherwise dealing with financial, personnel, and managerial issues rising from running their own IT development environment function. Despite some clear gains, there are, however, major restrictions to what a client can or cannot do within a cloud platform. The cloud platform may be beneficial for developers with a niche target to upload a configured application and run it within the platform's framework. And while there is more control here than with a cloud application, customers are nevertheless restricted to the platform's ability only and must fully recognize the difficulty in working 'outside the box. This layer is commonly referred to as 'Platform as a Service' (PaaS). (Crews, 2011)

Platform Providers

Instructors must 'not so tacitly' in considering what is essential to successfully run business software. Typically, we need hardware, operating systems, application software, databases, and server software. All these together constitute a platform and these are necessary to support an environment allowing business software to run effectively.

In attempting to set up a platform or searching for a suitable platform, it is essential to note different kinds of applications require different types of platforms. Some of the more well-recognized platforms include (Czarnecki, 2010) a Microsoft Platform (.NET application, Microsoft IIS, Microsoft SQL Server, and Windows Server), a Java Platform (Java applications, IBM WebSphere, Oracle database, Unix), and a 'LAMP" platform (Linux, MySQL database, Apache WebServer, PHP applications.) Thus early identification of a target platform, sometimes referred to as solution stacks, is essential. To complicate matters somewhat, some development tools are platform specific, such as Microsoft Visual Studio for Microsoft platforms. Other tools are more general purpose, such as Eclipse used by Java, C++, PHP developers, and others.

Sample platforms that provide much information online include Microsoft Azure and Google App, Engine, Amazon, and Force.com. See http://www.microsoft.com/windowsazure, https://appengine.google.com, Most sites allow the creation of accounts and provide limited access for experimentation.

Lastly, while there is no set of standards for PaaS, most platforms focus on particular domains, such as building social networks, support software development by programmers, and support non-programmers who wish to develop business applications

The Applications Layer

The Applications Layer, or top layer of the Cloud Pyramid, is the level where applications are run and often interact directly with a web browser. Here, applications are tightly-controlled and leave very little room for extension or modification.

This layer essentially contains applications offering web-based software as a service, and such services are often referred to as 'Software as a Service' (SaaS). Services offered to an client are no more than what the application can actually do. While applications are designed for ease of use, a client can only acquire the functionality available.

Application Providers

There are a number of outstanding software as a service providers. A few of the more notable providers include Microsoft, Google, and Salesforce.com. These suppliers typically host both applications and the organization's storage in the cloud. Using this model, the supplier licenses applications, and software used by many organizations is thus known as multi-tenant software – for clear reasons. Samples include email, word-processing, spreadsheet apps, presentations, collaboration tools for conferencing, real-time hat, document sharing, and business applications for Customer Relationship Management (CRM). See www.google.com/apps and Microsoft Office Live, www.officelive.com. Office Live is known as Microsoft Office 365 and has a number of productivity applications for small businesses.

5. MANAGING EXPECTATIONS IN REFRAMING THE CAPSTONE SEQUENCE

So how do educators leverage what is going on in the industry to reframe our capstone projects to provide students with a real awareness of what they may encounter in the real world? First of all, we must be realistic as to exactly what our students can do, will do, and what our expectations can / should be. Further, what are

the expectations for instruction to support such a venture? There is great complexity in setting up such a capstone sequence, as it is fraught with pitfalls. Yet, it seems appropriate that the capstone sequence oriented around cloud computing be an alternative instructional approach in meeting the spirit and intent of such a sequence.

Contact Providers to Discover Availability

The instructor needs to contact a number of cloud providers in order to determine exactly what offerings might be made available to students participating in a capstone sequence. It is likely providers might be willing to modify their offerings to educators as compared with individual offerings now commonly provided to those simply taking a look at their offerings. There are a number of providers with websites currently offering a variety of services to an interested individual. Yet these offerings have limitations.

General concerns would include:

- What software, services, support might be made specifically available;
- Would these be available to team / student use;
- How would these resources be made available and how managed;
- Are there tutorials, sample features provided with metrics, data constraints, application size constraints, performance constraints, number of accounts, contact points, and
- Are there sample reports of usage in the literature, and more.

The instructor also needs to determine the thrust of the course namely, would the class be undertaking development work in the cloud or using software provided as services (SaaS) by the provider or both. The answer to this question provides instructional alternatives.

A Changed Curriculum is Necessary

Undertaking this approach requires a sobering consideration as to expectations. It is imperative to consider simply what our students can do and what they will do. Exactly, what are expectations to be? While very proactive instructor dedication is crucial and sincere, hardworking students are also essential, these are not enough. Prerequisite / co-requisite courses may well need a facelift in order to support a capstone sequence centered on application development in the cloud.

A pre-required operating systems course that includes topics such as virtualization, service-oriented architectures (Haines, 2010), and cloud computing prin ticples is necessary. A networking course requiring development of a modest web-based application based on a model-view-controller architecture would provide good background. Hefty application programming and a strong database course need to be in the student's background.

Some of these may require changes; others not. But these are important to provide background and manage expectations of both students and instructors.

6. CAPSTONE PROJECTS, TEAM ORGANIZATION, DELIVERY, AND ASSESSMENT

One of the difficulties in organizing the content of a capstone course (or any technology-focused academic program) is maintaining a level of relevance to the challenges in today's globally-active workforce (Lesko, 2009) There are many proven successful approaches to capstone courses. Here are some additional recommendations.

Capstone Project: Two-Pronged Approach

It is well known that capstone sequences are taught in very diverse ways. Some tend to be very technically rigorous and emphasize analysis, design, and testing in very systematic ways. Others emphasize business concepts and project management skills and are sometimes found in MIS programs. (Hojdusz, Isola, Granger, and Gadayev, 2010) The approach offered in this paper provides benefits for both but emphasizes the former. Both approaches have strengths and weaknesses.

Capstone Using Saas. If the approach taken in the capstone sequence centers around SaaS, the team effort will center around investigating SaaS providers in an effort to establish the desired application. While the service is 'provided,' the team must still perform all the preliminary activities found in many projects along including activities such as analysis, requirements specifications, testing and more. Various providers need to be investigated by the student team. Deciding the best SaaS provider

given the requirements of the project and the services provided require a good deal of analysis. The downside of this approach is that software development itself (methodology selection, design (architectural and detail), programming, and additional activities typically undertaken in development projects are not undertaken. Yet building models of requirements, investigating providers, obtaining services, testing, deployment and related activities are undertaken and provide great value to programs oriented in this manner.

Capstone Using All Cloud Layers. For a capstone sequence centered on developing software by selecting an appropriate methodology, and undertaking specification, design, implementation, testing, and deployment, developing an application in the cloud provides unique challenges and opportunities. Can we require a student team or one with significant tech support to establish an infrastructure? This is very doubtful. To make this team effort more realistic, it is recommended the team address only platform issues and development issues with the assumption that the platform includes an underlying infrastructure. This is discussed in more detail in the next section.

Both of these two approaches, however, do introduce cloud computing into the curriculum in a meaningful way. By incorporating one of the two options in the capstone sequence, the importance and relevance of cloud computing is established.

Team Organization

Create Teams. Team formation experience has shown that there are usually a few students who are very technically competent and can form the basis for a team that could be used to develop a platform to support development. Such a team might have to work closely with the instructor, local technical support, or both. Realistically, establishing a student-led, infrastructure-platform team appears feasible.

Once the team for the infrastructure-platform is formed, the application team(s) undertaking software development (along with teams using SaaS to meet business applications) must be created in accustomed ways – student preferences, polling, skills, desires, and more. (LeJeune, 2008) The professor must prepare guidelines and specifications / constraints to guide these diverse yet specialized teams to manage expectations so that each team has a clear understanding of both their responsibilities and dependencies.

At this point, there appears to be two choices for the infrastructure-platform team: use a commercial platform or develop a private platform with technical support locally.

Commercial Platform. Regardless of the choice, the infrastructure-platform team will need to study existing platform models, providers, and characteristics. The team must research reports by platform providers and by clients of candidate existing platforms to become more aware of the strengths, weaknesses, and constraints of each. If the decision is made to use a PaaS from a provider, special care must be taken to learn about the chosen provider's other characteristics such as security, compliance, and perhaps legal constraints. Additional, this team must document a clear interface for the application team(s) that will be using this platform. The opportunity to investigate PaaS providers, work with these providers using the capabilities made available to the team, and interface with the teams developing applications in the capstone sequence by providing guidance would not have been possible in the more traditional capstone approaches.

Team Developed Platform. If the infrastructure-platform team is to develop a platform as part of the capstone sequence, establishing a set of development tools (solution stack) in a platform can be very time consuming, and keeping these plug-ins up to date is daunting administrative work. Administrators of such a platform require a significant competence in a variety of skills. This team must deliver a solution stack as a service. The stack must provide facilities for complete developing, testing, hosting web applications, database services, network management, and other services. Thus an advisable close relationship with the departmental technical support is needed to build, test, and validate the platform.

The major effort by this team will be designing a base set of facilities for potential developers. Work by the team must involve establishing and thoroughly testing all the facilities constituting the solution stack, such as ensuring solid, reliable database connections, network connections, and any other anticipated user needs. The interface must be both useful, learnable, and documented.

Delivery and Assessment

Application Teams. While some application teams may be those using SaaS, other application teams may be those using the platform solution stack developed by the infrastructure-platform team or hosted by a commercial provider. In either case. The provided platform is in direct support of the software development projects customarily accommodated in the capstone sequence. The application teams are thus the teams that will develop an application that may ultimately be offered as a service. If this is an additional goal, then those services offered by several of the main SaaS providers must be thoroughly investigated and incorporated into the application design effort. This would certainly extend the development project to not only meet accustomed requirements but also to establish the project as a service. All of the traditional standard elements to the capstone sequence come in and are emphasized.

All of these development efforts must be frequently presented in the classroom setting. If the nature of the capstone course allows development of SaaS, then the presentation of experiences of teams undertaking Saas with existing providers must be presented. Similarly, the application developers who are using the platform provided by the infrastructure-platform team or platform providers must also be presented. These deliveries must include how each of the traditional software development activities was impacted by the nature of the capstone team's project. These objective assessments will form the basis for an evolving improved process of the capstone sequence in the cloud.

7. CONCLUSION

There is little doubt that cloud computing is making inroads to many areas of technology. Thus it seems only fitting that students be able to develop software in the cloud. A series of recommendations are suggested in this paper based on a generalized appreciation of capstone courses. The wide diversity of capstone course formats offers a number of ways in which cloud computing might be incorporated into the capstone sequence. Regardless of the capstone approach using cloud thinking, a post mortem assessing the pros and cons of any approach is encouraged in an open forum Since the recommend approach is likely a departure from the more traditional ways of conducting senior capstone courses, it is anticipated that there will be a number of unforeseen problems. But just as we have evolved in our methodologies to develop software using iterative approaches with continuous assessment and the embracing of feedback, so too this approach - completely revamped with experience and lessons learned – can mature into a serious capstone sequence featuring development in the cloud. Hopefully, this is a starting point for some of us, as there is much to be learned.

8. REFERENCES

Bell, Michael (2008). Introduction to Service-Oriented Modeling: Service-Oriented Modeling*:* Service Analysis*,* Design*,* and Architecture, Wiley & Sons.

Bell, Michael (2010), SOA Modeling Patternsfor Service-Oriented Discovery and Analysis, Wiley & Sons.

Bryant, Robert, George Hauser, Greg Pleva "The capstone course: what are we looking for?" *Journal of Computing Sciences in Colleges*, Vol 17 Issue 2, December 2001

Clear, Tony, Michael Goldweber, Frank Young, Paul M. Leidig, Kirk Scott*, Resources for Instructors of Caps tone Courses in Computing, ACM SIGCSE Bulletin*, Vol 33, Issue 4, Dec 2001

Crews, Ken. (2011, Feb 15). Talk: Understand Why Organizations are Moving toward the Cloud and What will be the Impact to You. Presented to: The North Florida Rational Users Group, http://www.nf-rug.com, Jacksonville, FL

Czarnecki, Chris, (2110) Cloud Computing Technologies: A Comprehensive Hands-On Introduction, Course 1200, Learning Tree International

Haines, Marc N., How a Service-Oriented architecture may change the software development process, *Communications of the ACM,* Vol 53, Issue 8, August 2010

Hojdysz, Wojciech, Michael Isola, John Granger, Arthur Gadayev, "Managing Real-World Projects in Capstone Computing Courses", csis.pace.edu/~ctappert, 2010

Kennaley, Mark (2010). SDLC Beyond a Tacit Understanding of Agile, Fourth Medium Press ISBN 978-0-9865194=0-6

LeJune, Noel F., "A real world simulation technique for forming software development teams in a capstone course," Journal of Computing Sciences in Colleges, Vol 24, Issue 1, October 2008.

Lesko, Charles, "Building a Framework for the Senior Capstone Experience in a Information Computer Technology Program," SIGITE '09: Proceedings of the 10th ACM conference on SIG information technology education, October 2009

Walczak, Adam, Dana Gardner (2010) Cloud Computing Expo: Introducing the Cloud Pyramid Retrieved Aug 31, 2011 http://gemsres.com/story/jul08/609938/Sheehan1.png

Xiaolong Jin and Jiming Liu, (2004) From Individual Based Modeling to Autonomy Oriented Computation, Lecture Notes in Computer Science, vol. 2969, Springer, Berlin, 2004. ISBN 978-3- 540-22477-8