INTEGRATING INDUSTRY TRAINING AND SUPPORT IN THE LEARNING MANAGEMENT SYSTEM

Ziad Kobti, School of Computer Science
University of Windsor
Canada

Abstract

Teaching state of the art software engineering practices on advanced cloud platforms is a challenge in terms of costs and barriers associated in making such industry platforms available to students and in terms of the advanced training that is required to support the students on these platforms during a one term course. We report on a partnership between IBM Canada and the School of Computer Science at the University of Windsor that enabled the students both the access and training in the Bluemix platform through transparent integration into the LMS platform. The students gained much experience that enabled them to be placed in advanced and competitive internship positions after completing their course work.

Introduction

One of the key measures for a successful program is the rate of employment of students after graduation. It is a measure of the time it takes them after graduation to land a paying job in their field of specialization, and one can further add the degree of satisfaction they find in their career. Post-secondary institutions, such as the University of Windsor, through its Office of Institutional Analysis, conduct regular surveys and produce a number of statistics to measure the rate of employment of students after graduation.

In line with The Ministry of Advanced Education and Skills Development 2015/16 key performance indicators (KPI), institutions collect data through surveys and make their results publically available. For instance, the most recent published survey for 2015/2016 on the university's institutional analysis site was conducted in 2013 on graduates of Bachelors or First Professional degree programs. Graduates from each group of related programs were asked questions regarding their employment situation at 6 months and 2 years after graduation. For the purpose of the survey the *employment rate* is defined as the number of employed persons, including full-time and part-time employment, expressed as a percentage of the labour force. The labour force in this context is those persons who were employed, or unemployed but looking for work.

The data presented in Table 1 show the results of the survey of graduate employment rate by programs of study six months and two years after graduation. While the data in Table 1 account for all the programs within a program category, there is no specific measure for an individual program. The computer science programs include both cooperative education and traditional programs. A separate analysis indicates the graduates from the cooperative based education programs are able to find employment much sooner in their field. Programs with cooperative education better prepare students for the job market after graduation. Not all programs, however, have this option, and it becomes necessary for educators to find other means for preparing the students to be better prepared for the job market.

Table 1

University of Windsor 2013 Survey of Graduate Employment Rate by Programs of Study Six Months and Two Years After Graduation.

Program	Six Months	Two Years
Agriculture/Biological Sciences	91.70%	86.00%
Business & Commerce	88.90%	95.70%
Computer Science	88.90%	94.40%
Education	84.00%	93.10%
Engineering	84.60%	93.90%
Fine Arts	81.60%	82.10%
Humanities	84.80%	88.10%
Kinesiology/Recreation/Phys Ed	83.70%	82.80%
Law	93.60%	92.20%
Mathematics	80.00%	87.50%
Nursing	88.60%	99.10%
Other Arts & Sciences	90.90%	93.10%
Other Health Professions	*	*
Physical Sciences	*	100.00%
Social Sciences	80.70%	90.60%
Institution Rate	85.50%	92.20%

Source: University of Windsor (2018) MAESD KPI 2015/16

The School of Computer Science at Windsor, Ontario, Canada, introduced in 2014 a new professional master's degree program with the aim to fill a need in the market for skilled computer scientists. The Master of Applied Computing, since its inception, enjoyed a significant increase in its enrolment almost entirely composed of international students. The program involved a series of courses provided over three consecutive terms, followed by a four or six month fulltime internship component. From an educator's perspective the challenge is not in the ability to deliver the courses, rather, it is in the success of the students in landing high quality internships and the satisfaction of the employers and the student interns with the placement. The real test to an education program is, after all, measured in its ability to properly prepare a graduate student for the job market. The more cutting edge the job type and the reputation of the place of employment that the students land for their internship, the higher the quality of the program itself. Furthermore, since the internship is the final component of the program, the students are not mandated to return to campus once they complete their courses. This is even more attractive to employers who like to hire their interns into full employment after their internship term concludes.

This paper describes a course delivery approach that integrates industry partnership into the classroom. The learning management system presents the fundamental platform that enables communication access between students, instructor, graders, and external industry experts. Through careful orchestration of course contents, industry training materials and assessment methods, the student experience was enriched with advanced technical skills that were seamlessly integrated into the course syllabus. Students progressing through the course gain both fundamental and current job trainee type of skills that would otherwise be only made accessible to new employees. This approach positively impacted the internship's placement rate and its quality.

The objective of this paper is to present a summary of the pedagogical innovation in a first term software engineering course and to highlight its impact on student internship placement. The rest of the paper is organized as follows: in the next section the course content and standard delivery approach are presented; next the integration of industry is discussed, followed by a discussion on findings and measurement of outcomes. Finally, the paper concludes with a look into the future as industry continues to evolve.

A Course in Software Engineering

Designing a new university course is a daunting task for any professor. There is no shortage of challenges from deciding on a textbook, to determining the scope and depth of the contents. The course learning outcomes must be clearly defined along with their specific assessment methods, and of course they must align with the approved program learning outcomes. A new graduate course is even more challenging due to its depth in the quality of the contents on the topic been taught. The instructor must present the state of the art on the subject matter and its current form as practiced in industry along with the latest research findings on the topic. A typical university professor who would be tasked to deliver such a course is not necessarily, in fact not likely, to be employed in industry given the time and contract limitations in most education institutions. Transferring such current knowledge to the student would involve the collection of recently published journal articles or in the case of industry, most up to date publicly released technical papers.

Consider the textbook choice where a newly released textbook on the subject matter would be at least two years in the making. Its contents are current in terms of reference to the past few years of knowledge that was culminated in the text. While such a textbook would be a great asset for many disciplines, in computer science however, a new version of a software platform is released in a relatively very short time cycle, rendering the previous version obsolete or unsupported. Take for instance mobile development on the Android or IOS platforms. For instance, while developers are learning to write applications for the IOS 10 which was released on June 13, 2016, IOS 11 was released on June 5, 2017, less than a year later. This is not to mention all the important updates and bug fixes released in between these two major releases. A textbook on Android or IOS would always be obsolete upon its release, or at the very least, lacking of many technical updates (Wikipedia).

A search query on amazon.ca for textbooks on “IOS programming” would reveal over 1000 results with majority of books predating the IOS 11 release date. It is a losing battle for a university professor to settle on a current textbook for such agile technology. The real challenge remains: How can the students be trained with the latest technology and be ready to hit the ground running , so to speak, at their employment placement?

Software engineering is in itself a broad discipline of study that, while relatively new compared to the traditional disciplines, has gained unprecedented popularity over the past few decades aligned with the exponential growth of computer users around the world. A well rounded education program would ensure the student has satisfactory aptitude and knowledge of general concepts in the software development process.

A typical generalized software engineering course would cover various topics on the software production cycle, including requirement analysis, design patterns, agile programming practices, software testing and verification, and other security and deployment considerations. Coupled with the lecture style discussions are lab based exercises that engage the student to develop software through a series of hands-on exercises and assignments. A major project is common in graduate courses where a team of students work together on a relatively major development and demonstration of a software product as a proof of concept on the latest technology platform. Here lies another challenge: which platform would one adopt in a classroom so as to ensure students work on a stable, relevant, state of the art, and, most importantly, well documented and supported platform. The choice of programming language becomes secondary to the choice of the platform architecture. Often, a software product requires the use of multiple programming and scripting languages to bring it up to production levels.

The Latest Technology: A Moving Target

One of the most difficult aspects of teaching advanced software programming remains to be mainly in debugging, particularly using the latest technology platforms, whether it is a combination of a new platform or compiler. Programmers spend the majority of their time debugging the code after spending relatively less time writing it. This is a documented fact in early classical software engineering projects, and, sadly, remains the case in modern software engineering projects. In spite of all the advancements and enhancements in tools and their capabilities, bugs remain in software projects.

Take for example a classical UNIX system where the programmer is to write some advanced scripts to automate some tasks for the users. While there are well known scripting languages, say C-Shell or K-Shell for instance, we identify stark differences between different versions and system implementations. For example, an HP UNIX does not handle commands the same way a Digital Alpha Server would. Some commands may not be supported while others are supported but require different usage because their implementation requires different parameters. Here lies the challenge to the programmer to identify these differences that deviate from the published (and obsolete) reference manual, which is now a useless document that fails to provide the programmer with accurate and correct instructions on using the commands for the particular underlying system and specific version installed.

Cloud Computing

The cloud computing architecture is taking the world by storm. The days of writing software for a desktop platform are long gone. In recent years, with the ubiquity of the Internet and particularly its speed and reliability enhancements, modern software development often involves a multi-machine or distributed platform. A *cloud* in the context of software can be thought of as the collection of hardware distributed across some geographic locations that work collectively on providing a service to users.

(See "Harness the Power of IBM" (https://onthehub.com/ibm/) and "What is Cloud Computing?" (https://www.ibm.com/cloud/learn/what-is-cloud-computing).

Such commonly used cloud systems include storage systems such as Dropbox or Google Drive. Beyond storage, these services can be very useful and diverse, take for instance Google Mapping where a programmer can write minimal code to tap into their application-programming interface (API) and enhance the application being built with advanced mapping features.

Of course the risks for a software developer here are high. Consider the security aspect involving the collection and interception of data between the client and the cloud. This, however, is not of concern in this context where we would focus on the development and education. In order to teach the development of software that make use of this technology remains a big challenge. With the frequent updates of the cloud service, the software written by application programmers can break down if it is not updated accordingly to support the recent changes. The best source for documentation is no longer a printed textbook, it is now the most recent online documentation presented by the platform developers.

Microservices Architecture

Service-oriented architecture (SOA) is a loosely coupled architecture that provides various services due to cover the requirements of the organization and business. SOA aims to make the software more reusable (Newman, 2015) and is composed of some services that have been aligned with the business (Daya et al., 2015). SOA requires the services of a service provider as a style of architecture. In addition, SOA characteristics can be determined by the collection of principles, patterns, and criteria of its architecture. These characteristics can consist of the modularity, encapsulation, loose coupling, separation of concerns (SoC), reusability, and composability. Moreover, it can be thought of as a middleware solution that has been optimized to support the service assembly, orchestration, monitoring, and management.

In Microservices architectural style, the software application has been divided into several smaller services. These services are responsible for doing their tasks as well as possible. They also interact with the language-neutral APIs like REST. Furthermore, a service can be deployed as a separate entity on the platform as a service (PaaS) or can be a process for operation system. Moreover, they are able to alter independently, without any needs of customer intervention (Newman, 2015). The IBM Bluemix Microservices architectural style is used for developing applications. Bluemix provides platform as a service (PaaS) along with containers and virtual machines (Daya et al., 2015).

Agile computing methodology aims to reduce the complexity of planning while maintaining a key focus on the value of the customer and harnessing a positive climate for participation and collaboration (Stober & Hansmann, 2010). IBM Bluemix DevOps is capable of developing, tracking, planning and deploying the software as a software as a service (SaaS) (Daya et al., 2015). It works on the cloud and can support continuous delivery. It can handle the accessibility of requirement applications. After an application is built, it can be deployed onto the IBM Bluemix cloud platform. (Note, at the time this paper was being written the name of the platform was changed to IBM Cloud, a simple proof of how fast paced technology can be). In addition, it simplifies the movement of the code to running applications. It provides the ability to track and plan as another service making the IBM Bluemix DevOps capable of doing agile planning (Daya et al., 2015). The stories and tasks creation is supported by the IBM DevOps Track and Plan via using the tools of agile planning. By connection of the track and plan service to plans and code, the plans synchronize with the development team's progress.

From Industry to Classroom: Shifting Directions

Cost of technology in the classroom is a burden on the education system and a major challenge that educators have to take into consideration when designing the practical aspects of their course. It is for instance not feasible to expect a student to develop a class project on a commercial platform when the access to such a platform is cost prohibitive.

Now assuming the cost barrier can be overcome by an agreement between the industry partner and the institution, where students and faculty are presented with free limited access to the technology. What the next challenge would be is how to train students on such platforms especially if these are highly agile. The faculty member must depend on the training materials presented by the industry partner that are publically accessible. While textbooks are useful for presenting a historical review of the technology, they fail to meet their goal in being current manuals to assist users on a current platform.

Figure 1 illustrates how a typical university or educational institution would set up a computer lab. First the server is acquired and maintained to support the software configuration or platform that is desired by the unit to be used in the classroom. Next a physical lab is created with a number of workstations that are regularly supported and kept up to date for students to use. The students would use the software on the workstations and the development tools that are installed in order to access the server and work on the practical aspects of the course and the subject matter under study. This is a common and classical configuration found in many universities and institutions to this day. The main drawbacks of this platform are the fact that a technical support staff on hand is always required to deal with the ongoing maintenance of this lab and server. It is very common to see the software and hardware slide into obsolete or outdated versions due mainly to time and cost of upgrades. The frequency of upgrades is sometimes too high that a typical lab would end up being outdated and its maintenance costs could exceed the abilities of the institution.

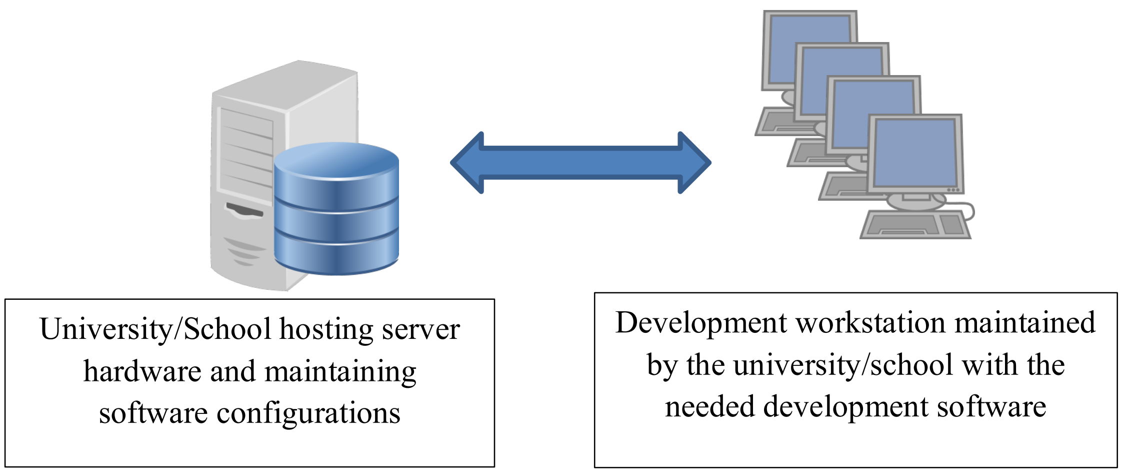
University/School hosting server hardware and maintaining software configurations
Development workstation maintained by the university/school with the needed development software

Figure 1. A traditional classroom configuration using traditional server client architecture.

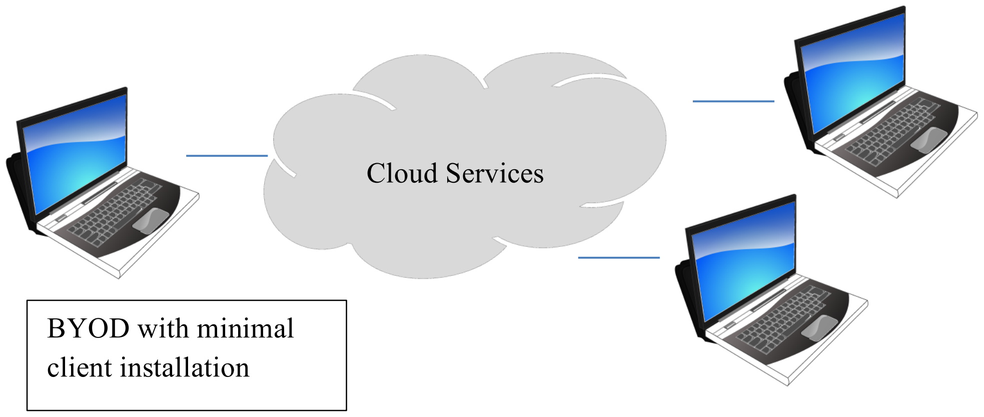

Figure 2. A modern classroom with no need for a centralized server and students bring their own devices (Bring Your Own Device – BYOD) that require very minimal client software installation and guaranteed to have the most up to date software pushed from the cloud upon connecting.

The new cloud technology platform enables a different type of access to services as presented in Figure 2. Here the server or servers are in the cloud. The student lab can remain either the classical classroom with workstations or following the trendier bring your own device (BYOD) lab. Here the institution invests in the purchase of services from the cloud and in the high-speed network connectivity and removes the costs of acquiring and maintaining servers and workstations.

The drawbacks of the configuration presented in Figure 2 relate to the fact that the students are distanced from the cloud services in terms of support. When the students started using this platform in the classroom with the aim to get access to the latest technology of a certain product platform, there were serious drawbacks and complaints mainly due to the fact that the platform was frequently updated. That consequently triggered ongoing major upgrades and growing frustrations on the end users who had to frequently update their client software and, often, older hardware devices.

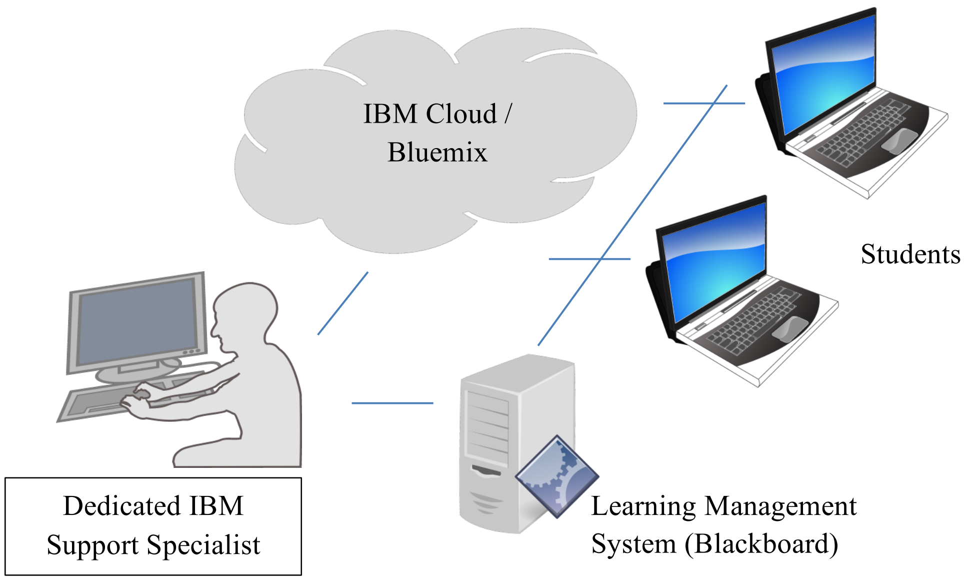
IBM Cloud / Bluemix
Students
Dedicated IBM Support Specialist
Learning Management System (Blackboard)

Figure 3. Extended modern classroom by means of seamless integration of the support specialist through the LMS to communicate with students and through the cloud to access code and assist in debugging.

The user frustration in the laboratory stemming from lack of support when dealing with an impersonal setting is a major challenge. Either the traditional educator is removed and students are expected to keep up on their own, or the traditional educator is present but he/she is as clueless to the changing behaviour of the system as the students due to the disconnect between the highly agile industry platform and the training materials presented to the trainer (or lab instructor). The ingenuity of the collaboration to address this problem is in the integration of a dedicated employee, a support specialist, who receives regular training and is within the company's inner circles and privy to the knowledge about all the changes occurring in the system. He or she is a seasoned programmer who is well versed and up to date on the platform's changing features. This person becomes a key in solving the problem of frustration from users due to the agility of the platform and lack of current manuals on the constantly changing technology. Calling or contacting such a person is equivalent to calling tech support on a product. The integrated approach helps avoid long holds or wait times for reaching a contact, having to deal with the service contracts, and adding frustration to students who would rather quit the project or perform poorly than have to pay for premium support services.

The partnership between the industry and the academic institution here flourished when, as shown in Figure 3, the support mechanism was integrated into the course learning management system (LMS). In this particular instance, Blackboard is used as the standard LMS that students and teachers have constant and easy access to. A special user account is created for the industry support specialist and connected strictly to the discussion forum as a moderator. Care has been taken that the support specialist does not gain unauthorized access to sensitive areas of the LMS that are normally only accessible to the instructor, such as the grading section. The specialist can now view the discussion forum, read student questions and respond to them while making the responses visible to the classroom users. It soon turns to a virtual classroom discussion on the LMS that connects the students to their instructor on one hand and to the industry specialist on the other hand. The student gets the best of both worlds: access to the instructor and access to tech support without the hassle of having to leave the LMS.

Teaching Classroom Setup	Traditional Classroom with the School maintaining server, workstations and all software (high cost of maintenance)	Modern Cloud Classroom with students connecting to the cloud based software (cost of software licenses to the cloud; no cost with IBM Academic Partnership agreement)	Modern with LMS Integration with students connecting to the cloud based software and a learning support specialist integrated into the course learning management system
Supporting Resources	Students use textbook resources that are out-dated or incompatible with the software installed on the machines	Students use up to date agile software on cloud resources and self-serve support	Students use up to date agile software on cloud resources and self-serve support
Student Training Challenges	Students complete the course with out-dated skills	Students are challenged with the new technology and frustrated due to lack of support and current resources in an agile development	Students are additionally supported with a live expert through the LMS
Job Readiness Rating	Low job readiness due to gap between software taught in the classroom and the latest industry software	Medium job readiness due to increased exposure to the latest technology but lack of mastery due to poor support	High job readiness due to high aptitude with the latest technology in current use by industry

Figure 4. Comparing different teaching approaches ranging from the traditional classroom to the cloud enhanced and supported model.

The progression of classroom technology and integration of the cloud, BYOD framework, and educators with industry is summarized in Figure 4 that presents a comparison of different teaching approaches ranging from the traditional classroom to the cloud enhanced and supported model. The comparison describes three different tiers of classroom laboratory setup that can be used by the educator and institution. The first involving a setup using a traditional classroom laboratory with server and workstations, which as previously mentioned would require high costs of maintenance. Next up we have access to a cloud platform thereby removing dependency on a local server machine, but that would come at high costs in terms of access to the cloud, unless an academic partnership is forged with industry, and in terms of student frustrations in accessing and gaining technical support on such a platform. The third tier is the targeted ideal where the cloud access is supplemented with integrated technical support right within the LMS platform that the students are accustomed to in their educational institution. Access to the learning specialist within the virtual classroom setting has proven to be a key feature for success of projects. As a result, when comparing the quality of the student projects between a cohort that had access to the specialist and those who did not there was a sharp difference in the level of achievement of the students as well as their satisfaction level with the course. The quality was measured in terms of the project deliverables when presented by the students. The more features they were able to encompass into their project and be able to implement and present, the higher the quality of the work. Furthermore, the end of term student evaluation survey which is a standard instrument for measuring the performance of the instructor and quality of the course overall were compared and there were clear improvement between the time the course was taught without the integrated specialist to the offering when the specialist was introduced. Students have revealed in the survey greater satisfaction with the course and more importantly better understanding of the subject matter.

Here it is important to link what was learned in the classroom to industry. After the completion of the course students are to undergo an internship placement with industry. The university employs a dedicated industry coordinator for the program whose task is to identify industry positions for the students. After the new integrated model was introduced, the coordinator reported back a major increase in the placement of the students in industry, particularly in high quality jobs.

It is worth noting that while students undergo the software engineering course and get initial training on the cloud platform they are also offered online training materials from IBM that if they were to complete they would receive a badge. In turn, students can mention that on their job applications and end up getting selected by industry for internship positions that were otherwise beyond their reach. Their technical training on the latest cloud platform presents the students with a more satisfying experience overall when measured in terms of the quality of training they received on the integrated cloud platform, the quality of the projects they were able to achieve within a course timeline, and the quality of the internship placement they land after the course has concluded.

Going back to the KPI mentioned earlier, it remains to be seen what institutional analysis would report in terms of students' employment rates after graduations. However, according to the high quality and increasing number of placements in internships it is possible to forecast a healthy increase in the employment rates and more importantly in the satisfaction of students with the education that they have received from the institution.

Conclusion

The rising cost of computer laboratory infrastructure and the increasing frequency of software updates present many challenges for education institutions to maintain up to date technology in their classrooms. Educators are presented with the challenges for maintaining current curricula and course contents, particularly training students in up to date software editions. The traditional computer classroom requires the maintenance of server and workstations up to date and in good operating condition, often requiring additional human resources and costs. In a closed classroom setting traditional textbooks fail to deliver accurate and up to date technical content for specialized software. In an advanced software engineering course it is critical for the students to have access to both current and accurate software documentation.

With the advent of the cloud computing and particularly software and platform as service technologies, clients need only a device capable of loading and running often minimal and thin client software to gain access to powerful and current resources that are constantly maintained on the cloud. IBM Canada, through their academic initiative enabled the University of Windsor students and faculty access to cloud services and training materials for enhancing student learning. This service while it provided students access to current industry software and the latest versions, presented several challenges in terms of the difficulty arising from using agile software. Its frequently changing versions and large amount of expectations on the students to read and self-train on several technologies presented some challenges and frustrations. To overcome this learning curve a learning specialist with high technical aptitude from the industry is integrated into the classroom's learning management system software. This enables a high quality learning experience where students are able to get their highly technical questions addressed, thereby mitigating the risks associated with the agile learning environment. The results of this practice unveiled their impact after monitoring the graduating class and their success in internship placement. An 84% placement rate of students in highly specialized and reputable industry institutions confirmed the high readiness of the students going into the job market.

Additional long term studies and enhancements will be considered in order to ensure this model remains sustainable and inclusive of other computer courses such as Internet and Database, and not just software engineering. Similar approaches to partnerships should be considered for other cloud platforms and services from the likes of Amazon, Microsoft and Google. For these to occur, the first challenge is to build the partnership between industry and academia to enable student and faculty access to technology.

References

Daya, S. et al. (2015). *Microservices from theory to practice: Creating applications in IBM Bluemix using the microservices approach* (1st ed.). International Business Machines Corporation. Retrieved from http://www.redbooks.ibm.com/redbooks/pdfs/sg248275.pdf

Newman, S. (2015). *Building microservices*. Sebastapol, CA: O'Reilly Media, Inc.

Stober, T. & Hansmann, U. (2010). *Overview of agile software development in Agile Software Development.* Berlin, Heidelberg, Germany: Springer.

University of Windsor. (2018). MAESD key performance indicators 2015/16. (2016a). Retrieved from http://www.uwindsor.ca/institutional-analysis/institutional-analysis/maesd-key-performance-indicators-201516

Wikipedia. (2018, March 20). iOS version history. Retrieved from https://en.wikipedia.org/wiki/IOS_version_history

Author Details

Ziad Kobti
kobti@uwindsor.ca