A Set of Best Practices to Design Face-to-face Teaching Sessions for Technology-centered University-level Computing Courses

http://dx.doi.org/10.3991/ijep.v4i4.4000

Ilse Baumgartner
Singapore Management University, Singapore

***Abstract*—Since more than a decade, all kinds of businesses and organisations are intensively exploring enterprise-level information systems to better integrate their business processes, information flows and people. Consequently, the industry demands for technically skilled, but also "business-savvy" IT professionals are permanently growing. To meet this need, more and more computing education programs try to incorporate enterprise-level information systems into their curricula. While there is some computing education research done to investigate the need for this new type of IT-business professional and to analyse general implications for higher education, only very few research works or practice papers exist which report on concrete attempts to design and deliver higher education computing courses which intensively use enterprise-level systems. In this paper, the author reports on a series of experiences made within the Bachelor of Science (Information Systems Management) degree program offered by the School of Information Systems (SIS) at the Singapore Management University (SMU). The primary focus of this paper is put on establishing a working set of best practices for the design of an effective structure of the face-to-face teaching sessions for courses which use enterprise-level systems and applications in their curricula. While this paper is principally based on education experiences made within the frame of an Information Systems program, the best practices presented in this paper are equally applicable to any other computing education field or even to the engineering education in general.**

***Index Terms*—Best practices, curriculum design, design of face-to-face teaching sessions, technology-centered university-level computing courses.**

I. INTRODUCTION

More and more organisations, businesses and institutions are attempting to break down the traditional silo-based working style by integrating their people, processes and information flows. An increasing number of organisations are working across functional and geographical boundaries – virtual teams, project-based structures, task-forces or teamnets involve people who are neither co-located, nor personally know each other, business processes are stretching across different departments and different responsibilities, and information is exchanged across different languages, different time zones and different geographic locations.

The traditional approach of delivering "silo-style" information systems does not support this highly collaborative and communicative environment. Consequently, the industry is increasingly looking for technically skilled, but also "business-savvy" IT professionals who are capable of designing, delivering and supporting such an integrative and collaborative information systems environment in an organisation.

This demand puts enormous pressure on higher education institutions.

To meet this industry demand, more and more computing education programs are attempting to incorporate enterprise-level information systems into their curricula. However, these attempts frequently lead to numerous challenges, from difficulties in linking the new content to the existing curriculum to the enormous amount of paperwork needed for the approval of a new course, track or major.

One of the most basic challenges experienced by the educators who attempt to use an enterprise-level system in a computing course is the never-ending quest for the "right" structure of the face-to-face teaching sessions of those courses. What is the "right" proportion of laboratory components in such a course? How long should a lecture part be? Or do we need a lecture at all? Is there any need for in-class exercises or tasks which are rather theoretical in nature? What is the right amount of project-based work in such a course – and does the face-to-face teaching session need to be concerned with this project work? Is there any need for theory-testing activities such as quizzes? And – finally – how much work should be done individually and how much work needs to be done in groups?

Contrary to professional education courses where the main focus in put on learning the mastery of a specific tool or system, university-level computing education courses need to establish an effective methodological and instructional chain in covering both, high-level concepts and theories related to organisational practices, organisational structures and organisational system implementation and practical hands-on abilities and skills in using enterprise-level systems and applications. Consequently, the educators are exposed to many challenges and difficulties in establishing an effective structure for the face-to-face teaching sessions of such courses and in trying to "link back" the theories and concepts taught in the course to the practical skills and abilities in using the enterprise-level systems.

While there is some computing education research available which investigates the current industry demands for this new type of IT-business professional and which offers an in-depth analysis on general implications of this increasing demand for higher education, only very few research works or practice papers exist which report on concrete attempts to design and deliver higher education courses which make an intensive use of enterprise-level information systems in their course curricula. Moreover, attempts to build a set of best practices in structuring the face-to-face teaching sessions for such technology-intensive university-level computing education courses are almost non-existent.

In this paper, the author reports on a series of experiences made within the Bachelor of Science (Information Systems Management) degree program offered by the School of Information Systems (SIS) at the Singapore Management University (SMU). The focus of this paper is put on describing the challenges related to establishing an effective and stable structure of the face-to-face weekly teaching sessions for courses which use enterprise-level systems and applications in their curricula. Using several courses of the program as examples, the author of the paper shows how a series of decisions related to the design of the face-to-face teaching sessions were made (and subsequently revised) when designing and delivering these courses. While this paper is principally based on education experiences made within the frame of an Information Systems program, the best practices elaborated and presented in this paper are equally applicable to any other computing education field - computer science, computer engineering, software engineering, information technology, or informatics – or to the engineering education in general.

The paper unfolds the following manner.

Section two of this paper undertakes a brief review of the existing research in the area of enterprise-level information systems use in higher education, particularly focusing on the lack of research and practice reports on design and delivery of courses based on enterprise-level information systems.

Section three briefly introduces the BSc (Information Systems Management) program at the School of Information Systems, Singapore Management University, and describes the current progress of the program in terms of embedding and using enterprise-level information systems in the program courses.

In section four, the author of the paper focuses on numerous design-related questions for all major components of the face-to-face teaching sessions. The subsection describing the lecture component particularly focuses on concerns such as the length of the lecture activities, the nature of lectures, the positioning of lecture activities within the structure of the weekly sessions, and linkage of the lectures with other activities of the face-to-face sessions. The subsection describing the laboratories and hands-on tasks seeks to share selected insights on such important topics as the nature of laboratory sessions, length of laboratory tasks, linkage of laboratories to the lecture components (and other components of the face-to-face teaching sessions), challenges with regard to the infrastructure required to execute laboratory tasks, and the most basic question of moving the laboratory components to the students' time outside the face-to-face sessions (i.e., letting the students to execute the labs individually outside of the usually class hours). In addition to the lecture component and laboratories, section four also provides selected insights with regard to embedding in-class theory-related activities, theory-testing activities, projects and other course components into the face-to-face teaching sessions.

Based on insights and experiences reported in section four, section five of the paper builds a set of best practices and presents three structure models for face-to-face teaching sessions of technology-intensive university-level computing courses.

The paper concludes with a series of recommendations for those university-level computing educators who are determined to design and deliver undergraduate level information systems courses involving enterprise level software packages.

II. LITERATURE REVIEW

There has been some research to understand the requirements towards "enterprise-system-literate" graduates and to describe the conceptual integration of enterprise-level information systems into higher education curricula [1] [2]. Interestingly, the research is principally focusing on the integration of ERP (Enterprise Resource Planning) systems into the higher education curricula [3] [4] [5] [6] [7] [8], while other types of enterprise-level information systems are heavily underrepresented.

While some of the research questions the motives for the adoption of enterprise-level information systems (mostly ERP) into higher education curricula [9] or presents a high level analysis of costs which will be incurred by universities when adopting ERP systems (such as SAP) for their curriculum [3], most of the education research in this area offers some conceptual analysis on a high level curriculum design and implementation.

Wang and Hwang [6], for example, present a framework of innovatively integrating ERP into multiple core and elective courses proposed in IS 2010. Atif et.al. [10] present a model curriculum that prepares students for supporting large Enterprise Information Systems (the presented model curriculum is principally based on ERP). Swanson and Helpner [11] propose in their study a knowledge management framework for developing and managing ERP curriculum within business schools.

Due to the complexity of the systems, there are obviously difficulties and challenges to be expected when attempting to deploy and use enterprise-level information systems in a course, particularly in large, compulsory courses.

Wagner and Pant [12], for example, discuss in their teaching note the practicabilities and challenges encountered in a course which teaches enterprise-level database management systems, particularly focusing on the use of a server virtualization tool that is commonly applied to allow students to gain experience in using several of the most popular enterprise-level database management systems (DBMs). Davis and Comeau [13] describe the design, delivery, and outcomes of a course on enterprise integration at the senior undergraduate level in an e-business track. The usefulness of insights described in this paper is, however, limited for computing educators as this paper describes a course with a clear management centricity (instead of technology centricity).

As shown above, there is some research done in the areas of high level enterprise systems curriculum review, design and recommendations. However, there is an obvious shortage in research reporting on actual implementation scenarios – i.e., papers or works reporting on the actual use of enterprise-level information systems in the course, focusing on educators' experience, students' experience, tutors' or assistants' experience and making practical recommendations and suggestions to institutions or educators who are interested to explore the use of enterprise-level information systems in their course curricula. Moreover, there are almost no contributions available which would attempt to extract and present some more general best practices in designing and delivering technology courses which use enterprise-level systems and applications.

III. ENTERPRISE-LEVEL INFORMATION SYSTEMS IN THE BSC (ISM) PROGRAM AT THE SCHOOL OF INFORMATION SYSTEMS, SINGAPORE MANAGEMENT UNIVERSITY

To produce IT skilled, but also business-savvy graduates, the computing education programs must expose their students to the design, implementation and support of enterprise-wide systems and applications. A typical undergraduate program in fields such as information technology, information systems or software engineering is built in a staggered fashion, covering the IT fundamentals in the initial years of the studies and moving into more complex topics in the senior years. Most importantly, each of the information technology topics is covered in an isolated manner – data storage and databases, programming fundamentals, project management, networks or information security etc. These isolated courses are, thus, not suitable to expose students to a complex, integrated, interconnected information systems environment where all those components – data management and storage, network, security, application design and others – need to be combined in order to produce business value for an organisation and support its business strategy.

Currently, most computing program graduates still acquire skills related enterprise-level information systems on-the-job. To change this, many higher education institutions strive for new education models in order to provide a comprehensive coverage of best practices needed to make IT graduates effective professionals in a fast-changing IT market.

Most frequently, the introduction of enterprise-level information systems has happened within business schools' curricula [14] [15] [16]. However, principally, enterprise-level information systems need to be viewed as a native computing (i.e., Information Technology, Information Systems, Computer Science) domain, which means that computing programs need to produce graduates who understand the full lifecycle of an enterprise-level information system and who are able to make a successful use of both, organisational assets and technology assets, to create value for the organisation.

Since most of the organisations are adopting enterprise-level information systems within and across organisational boundaries, enterprise-level information system education has to comprise architectural thinking, systems thinking, and systems governance processes. Thus, the enterprise-level information systems education needs to be cross-course or even cross-track based.

The BSc (Information Systems Management) program at the School of Information Systems (SIS), Singapore Management University (SMU), is built upon three foundation pillars, namely: the application of information technology in the context of business, exposure to real-world business problems and processes through business scenarios and cases, and cross-training in business, management or social sciences though options such as courses in other schools or 2nd majors.

As of now, several core courses of the program use enterprise-level software in their curricula.

The Process Modelling and Solution Blueprinting course (PMSB) is a second year core course. To execute the lab components and the projects of the course, students use the IBM WebSphere Modeller, a comprehensive enterprise-level business process modelling and development environment that enables PMSB students to model business processes and simulate and analyse proposed improvements on the company's business operations. The PMSB course teaches the students concepts and methodologies required to translate business process change requirements into effective IT solutions.

The Enterprise Integration course (EI) is a second year core course. The EI course principally focuses on concepts and technologies which are required to design and implement enterprise integrations solutions (using the paradigms of Service Oriented Computing and web services). TIBCO BusinessWorks is the principle enterprise-level tool used in this course. The students are employing this particular tool to expose systems as services, to build new services, and orchestrate and assemble services into applications.

The Enterprise Web Solutions (EWS) course is a third year core course and it exposes senior undergraduate students to enterprise-level system design and implementation (using enterprise portal technologies as a sample framework and platform). Contrary to the EI course and the PMSB course (each of which uses one main enterprise-level tool), the Enterprise Web Solutions course uses a comprehensive package of enterprise-level systems and tools – such as Microsoft SharePoint, Microsoft SQL Server and Microsoft Search Server. Moreover, students are also exposed to several enterprise-level IDEs and other tools in this course.

There have already been several attempts to coordinate among the courses using enterprise-level information systems – particularly with the aim of exposing the students to different aspects of the same enterprise-level system or application from different perspectives. Some of the courses have attempted to use one single case study across different courses, some other courses have explored a possibility of implementing a capstone project of one course as a direct extension of a capstone project of another course (using the same enterprise-level technologies and applications). These integration efforts, however, are usually associated with considerable overhead concerning the preparation of the technical infrastructure, as well as with difficulties in adapting the course curriculum in order to make the integration efforts transparent for the students and productive for the teaching staff.

As of now, most of the experience and insights has been shared in the area of designing face-to-face teaching sessions for the courses. The following section presents those insights and attempts to develop three high-level structure models which have been successfully applied across several courses of the program.

IV. COMPONENTS OF FACE-TO-FACE TEACHING SESSIONS: DESIGN AND IMPLEMENTATION

A. Introduction

There are several instructional methods which represent essential components of face-to-face teaching sessions for almost every university-level computing course – such as lecturing components, hands-on or practical components, theory-testing components, in-class activities or exercises components and others. Contrary to a "pure" professional or vocational IT course, only a few of those components may be directly concerned with the practical skills, abilities and competencies, attempting to train the students on how to use (or support) a particular enterprise-level system or application. Most of the effort is, however, naturally put into building a higher level understanding of theories and concepts which those enterprise-level systems are built upon (i.e., instead of *how* rather focusing on *why*). Nevertheless, there must be an effective "closure of the loop" in place where the high-level concepts taught in the course are clearly mapped to the technical skills and capabilities acquired by the students in the practical exercises.

Conceptualising, designing and implementing this "closure of the loop" is, however, anything but trivial. Many factors make this task difficult: the time constraints of the face-to-face sessions, number of students in one particular session, the complexity of the chosen enterprise-level system, students' previous experience and exposure to any enterprise-level systems etc.

One of the most difficult questions to answer when designing a course which extensively uses enterprise-level software is the role and weight of every single instructional method within the particular face-to-face teaching session where this instructional method gets employed.

B. Lectures

The most important question regarding the lecture component is the *principle decision* of embedding lecture-like activities into the structure of the face-to-face teaching sessions. Since lectures are considered to be one of the basic teaching methodologies in university-level education, the decision of heavily reducing (or even eliminating!) the lecture component in the face-to-face teaching sessions is a difficult and controversial one.

The author of the paper has been extensively experimenting with completely removing face-to-face lectures and replacing them with video-based learning material which the students are required to view and independently absorb BEFORE the face-to-face teaching session.

Several important insights emerged here.

Firstly, although there was a considerable gain achieved in terms of timing (making more than one hour per face-to-face teaching session available for different activities in class), a large fraction of this class time was actually spent on answering diverse questions from students concerning the video material (there was also a need to offer additional consulting hours for those students who wanted to clarify their questions upfront, before coming to class). The students seemed to struggle in understanding even comparably simple concepts covered in the video – when being deprived of the possibility to ask questions when watching the video.

Secondly, it was difficult to establish a common and shared understanding of some of the concepts covered in the videos. The range of interpretation was usually very wide (and, frequently, the students were interpreting one and the same concept completely differently).

Thirdly, particularly students with no or limited previous exposure to enterprise-level systems, had considerable difficulties in grasping the basic complexities underlying the implementation of an enterprise-level system. Those students usually tended to seek for personal consultations with the teaching staff.

And, finally, the overhead in preparing (and constantly keeping up-to-date) the video material was considerable. The video material was mainly usable for up to two or three terms. Since the video materials for all teaching sessions were naturally linked to each other, changes in the video material of one session enforced changes in all subsequent video materials. Constantly changing and re-recording videos caused an organisational overhead which was initially largely underestimated.

Considering the experiences highlighted above, embedding a lecture component into the face-to-face teaching session seems to be justified. But what is the optimal length of a lecture component? Moreover, is it meaningful to split the lecture component into several parts – or is it more productive to have one single lecture component? And where with regard to the structure of the face-to-face session should the lecture material be covered – at the beginning of the session, towards the end, or rather somewhere "in-between"?

Lecture components in a course dealing with an enterprise-level system implementation will mostly be concerned with explaining to students selected theories and concepts which are essential for the selected system or application. For example, if a course is concerned with the implementation of an Enterprise Resource Planning (ERP) system, there will be a necessity to expose the students to concepts such as business process management, service oriented architecture, transactions, composite applications and others. Most of those concepts, however, are directly linked to specific hands-on skills or practices which the students need to acquire in the course.

Based on many experiences in delivering courses focusing on enterprise-level systems, the author of the paper sees clear advantages in using a "mini-lectures" approach – splitting the lecture component into several parts and delivering those parts alternately with hands-on or laboratory components. The principle advantage here is the opportunity of constantly reinforcing the linkage between the theory and practice and clearly showing to students the applicability of all theory concepts to the practical skills. The basic approach here would be the following: having an introduction lecture (e.g., 20 minutes), then delivering a short "mini-lecture" (e.g., 15 minutes) on a specific theory or concept, having the students to accomplish the associated practical hands-on task (e.g., 15-20 minutes), have a short "wrap-up" discussion on this component (e.g., 5 minutes), deliver next "mini-lecture" on a specific concept (15 minutes), continue with students accomplishing the next practical task and so on. During one face-to-face session, the students seem to successfully absorb up to three such iterations.

Generally, the author of the paper recommends the total maximal length of the lecture component not to exceed ⅓ of the total length of the session (e.g., 1 hour lecture session if the total session is 3 hours), with the remaining time of the face-to-face teaching session to be spent on other class activities (such as laboratories, in-class activities, testing activities etc.).

C. Laboratory and hands-on tasks

As discussed above, the use of enterprise-level systems in university computing education is currently becoming more and more mandatory. Incorporating laboratory (or similar hands-on) components into the face-to-face teaching sessions seems to be the only feasible way of practically exposing students to those systems.

However, as already indicated above, one of the most important issues to address here is the linkage of the hands-on tasks with the concepts covered in class. A possible approach to implement such a linkage would be delivering the lecture component alternately with the laboratory component.

One major issue with this approach might, however, be the necessity to split the laboratory component in order to design small chunks of hands-on tasks instead of designing and using one coherent laboratory session. For some of the courses, the author of the paper has observed the tendency of students approaching the laboratory components in a rather mechanistic way instead of trying to understand the broader context of the laboratory exercise. This seems to be a particular issue in courses extensively using case studies in their teaching process. Case studies are necessarily leading to larger and more coherent contexts for hands-on and laboratory tasks, and in such a situation the use of one large, continuous laboratory session seems to be more productive than isolated, small hands-on sessions referring to specific concepts or theories.

Another problematic issue when using this lecture-laboratory-alternate approach might also be a very different speed which the students have when completing the laboratory tasks. This usually leads to some students completing the tasks much quicker than others and being forced to wait until the entire cohort completes the exercise. Frequently, this situation causes dissatisfaction of more advanced students since they are bored and feel that the time is not used productively. One way of dealing with this is to expose the more advanced students to problem exercises, advanced training etc. while they are waiting for their colleagues to complete the basic tasks (which, in turn, might be very distracting for them if – after the slower students finally complete their exercises – the teaching staff member wants to continue with the lecture component without permitting the more advanced students to complete their additional exercises).

The approach of having one single laboratory component has, however, its merits, too. Firstly, as observed by the author of the paper, a large coherent laboratory component allows the students to more easily grasp contexts of more complex implementations, implementations which involve more than one system, implementations which require students to use techniques which they have learned across several teaching sections, and implementations which require students to think of "why" instead of just "how". When using one single laboratory component per face-to-face teaching session, it is also possible to extensively use larger case studies to establish a clear context for the laboratory tasks.

When using the "single-laboratory" approach, the lecture – necessarily – needs to be delivered as one single block. The primary question here is the following: if there is only one lecture component and one laboratory component, in which sequence do we deliver those components? Does the lecture go first, or should the students first be exposed to practical tasks (which illustrate the concepts covered in the lecture) and then have the lecture delivered?

Two contrary arguments could be presented here: (1) it makes sense to deliver lecture first to give the students the basic understanding of the concepts they will actually touch upon in the practical session (otherwise, how should the practical task be accomplished without having any idea of what this practical task is actually attempting to illustrate?), however, (2) letting students to experience the concepts "in action" before actually explaining those concepts to them makes this explanation far more concrete and tangible since after having completed the laboratory component the students already have a good sense of what the professor is actually talking about.

Based on their experience, the author of the paper tends to favour the first approach – delivering the lecture first followed by the practical tasks. This approach, however, seems to be only appreciated by students if the lecture component is embedding at least some practical demonstrations showing the relevance, practical implications and actual use of selected concepts taught in class. A complete lack of any practical demonstrations or practical references usually tend to lead to a much large timeframe needed for students to fully understand the context of the laboratory component and to clearly link it to the concepts and theory taught in class.

The final approach to the laboratory exercises which the author of the paper would like to briefly discuss is moving the hands-on tasks "out" of the face-to-face teaching sessions and instructing the students to execute the laboratories independently (i.e., outside of the class time). Particularly courses using more than one enterprise-level tool and exposing students to complex implementation scenarios frequently lack sufficient time to (fully) perform laboratory exercises in class during the face-to-face teaching sessions. In such cases, video-based laboratory exercises completed outside of the class time might be used – provided that the students receive sufficient explanations and guidance on the underlying principles of the laboratory tasks – and are able to independently establish the linkages between the laboratory tasks, the lecture material and any theory-based in-class activities. The drawback of such video-based and out-of-class laboratory sessions (although they are very frequently well-received and appreciated by students, particularly due to their flexibility, allowing students to perform laboratory tasks in their own time, at their own convenience and at the place preferred by them) is the considerable overhead for the teaching staff to prepare them. Planning, designing and recording video-based laboratory components is very time consuming. Since – frequently – laboratory components in courses using enterprise-level systems are linked to each other and are referring to each other, changes in one laboratory component enforces changes in other components, too. Videos (and the corresponding documentation, instructions etc.) need to be re-designed and re-recorded – which, in turn, results in considerable time investments for the teaching staff.

D. Theory-based exercises, tasks, assignments executed and completed in class

Lectures and laboratories are considered to represent the foundation of technology-centered computing courses (and are widely accepted instructional methodologies for such courses). It can, however, be argued that – particularly when attempting to teach complex enterprise-level implementations (usually involving more than one single stand-alone system and usually dealing with complex organisational scenarios) – many students still seem to have considerable difficulties in establishing the linkages between the "high-level" theory delivered through the lectures and the practical "low-level" hands-on skills taught in the laboratory components. The "gap" between the lecture and the laboratory is still considered too wide to be able to effectively link those two components.

One of the approaches in helping the students to establish this linkage might be the introduction of theory-based (but still to a certain extent "hands-on") in-class exercises. Such exercises (or tasks, or assignments) primarily focus on selected concepts covered in the lectures and require students to produce and submit deliverables (e.g., documentation, templates, diagrams etc.) for tasks specifically formulated based on the material covered in the lectures. With this, such in-class exercises have proved to be an extremely valuable instrument in establishing an "intermediary stage" between "pure theory" and "pure practice". Such in-class exercises would usually be positioned between the lecture component (mostly a "single-lecture" component) and the laboratory component (most frequently a "single-laboratory" component). An example of such an in-class activity would be, for example, designing a workflow diagram for a given business process to be finally implemented as a software component in an enterprise portal solution. This particular in-class exercise is based on theory material covered through the lectures (business process management, automation of business processes, concept of workflow, state machine vs. sequential workflows) and provides an intermediary step towards the practical laboratory task (which is coding and deploying a custom workflow component to an enterprise portal instance).

E. Projects

The capstone projects or final projects usually represent the principle means for students to showcase the knowledge and skills gained through the course. For courses using enterprise-level software and packages in their curricula, final projects are essential components which – usually – account for a comparably large faction in the course assessment structure (30- to 40 percent is not unusual number). Consequently, the most basic question which the course designers and deliverers have to answer is the necessity to devote any time of the face-to-face teaching team sessions to the final projects of the course. Is there a need to be concerned with the project during the class time – or should the project be completely moved out of the class time? If any class time is devoted to the project – from what kind of project-related activities the students would benefit most?

Based on their experience in delivering courses which use enterprise-level systems, the author of the papers strongly suggests that only minimal time of the face-to-face teaching sessions should be concerned with the final or capstone projects of the course. Instead, the design of the face-to-face teaching sessions should be effective enough to enable students to grasp the linkages between the underlying theory and the actual practical tasks and to effectively (and independently) apply those linkages to the final or capstone project.

F. Quizzes and similar theory-testing activities

Although university-level computing courses extensively using enterprise-level systems and applications in their curricula are mostly focused on understanding the technical design of such systems and analysing the organisational impact of such implementations, these courses are nevertheless more than vocational or professional training units. Students of such courses are expected to acquire far more than practical "hands-on" skills in using (or implementing, or supporting) such systems. Therefore, a large part of such courses is usually devoted to the underlying theory and concepts accompanying large organisational implementations. Consequently, there is a natural question for the designers of university-level computing courses extensively using enterprise-level systems and applications in their curricula: does it make sense to embed any theory-testing activities (such as e.g., quizzes) into the face-to-face teaching sessions?

The author of the paper has been extensively using different styles of theory-testing activities in class – such as pop-quizzes (i.e., quizzes which are not announced in advance), regular quizzes (i.e., quizzes which follow a pre-announced schedule), short-answer-exercises (i.e., 10-15 minutes long theory-testing exercises asking students to provide 1-2 sentences answers to specific questions), in-class discussions, self-assessment exercises and others. However, these activities do not seem to particularly contribute to the students' ability to build the linkages between the theory covered through the lecture components and the hands-on skills covered through the laboratory components. For this reason, the author of the paper very recently has started to use (optional) non-graded self-assessment-style theory-testing activities performed outside of the class time.

V. BUILDING A SET OF BEST PRACTICES

Enterprise-level information systems are not only about technical competencies. Rather than that, a course using an enterprise-level information system focuses on the underlying processes, it depicts involved people and it talks about the business value which this system (or a set of integrated systems) is delivering to the company.

Thus, a higher education computing course which chooses to use enterprise-level information systems or tools in its curriculum must necessarily offer more than a vocational or professional training: in addition to getting the students to practice and use such systems in a "real-world"-like technical environment a clear linkage needs to be established and shown between those systems and the organisational capabilities, structures and strategies.

In the following section, the author of the paper would like to suggest three possible models of structuring the face-to-face teaching sessions for technology-centered undergraduate computing courses.

The first model (figure 1) represents the most basic approach to the face-to-face teaching session design for courses extensively using enterprise level systems in their curriculum. A single lecture component is used to establish the links between the theory covered in the course and the hands-on tasks executed in the laboratory component. There are no iterations used in this approach.

While this approach seems to be the easiest to design and implement, this model can be recommended for relatively simple scenarios only. This model will work most successfully when exposing students to comparably simple concepts related to enterprise-level system implementation – and in situations when working with students who already have had some exposure to similar enterprise-level systems and applications in other courses of the program. When using this model, there is no particular need for a particularly detailed explanation or reinforcement of concepts.

The second model (figure 2) represents the lecture-laboratory-alternating approach which enables more frequent and more granular reference to high-level concepts taught in the course. The author of the paper suggests up to three basic iterations when using this model.

This model will be most successful in scenarios when there is a need to introduce students to completely new concepts or theories (which, in turn, lead to a set of new skills and abilities which the students need to acquire). In addition, this model will be most useful in situations where the linkage of the underlying theories to the practical skills can be explained in a staggered fashion only – as one concept is building upon another concept, and one practical skill is based on another practical skill.

For example, in a course using an ERP system, there might be a need to explain to students the concept of ERP transactions. Some of the most important concepts to demonstrate here would be the role master data, followed by Bill of Materials (BOM) and, finally, transactions. Using the second model would be the most appropriate way of proceeding here – first, having a short "mini-lecture" on master data, followed by a hands-on laboratory task asking the students to create material master data, then moving to the next "mini-lecture" explaining the role of BOM and having a short hands-on exercise to create a BOM based on the initially created material master data, and then – using lecture 3 and hands-on task 3, demonstrate how those concepts are linked to the concept of transaction in an enterprise-level system.

The third model (figure 3) returns to the basic approach of having a single lecture and a single laboratory component. However, to facilitate the process of establishing the linkages between the conceptual part of the course and the hands-on skills taught in the course, in-class activities and exercises are introduced. These exercises are directly built upon selected concepts covered in the lectures and are representing an intermediary stage in linking the theoretical concepts taught in lectures and practical skills and abilities taught in laboratory components.

This model would be primarily used in situations where the nature of the course and the nature of the enterprise-level system used in the course require broader contexts. Moreover, this model will also be used in courses which are principally based on case studies.

For example, in a course which uses an enterprise portal server package to teach students implementation of enterprise-wide portal solutions, there might be a need to explain the concept of portal topology (i.e., the basic structure of the enterprise portal.). Design of a portal topology typically represents a complex, multi-step process which consists of two main phases – firstly, designing the initial

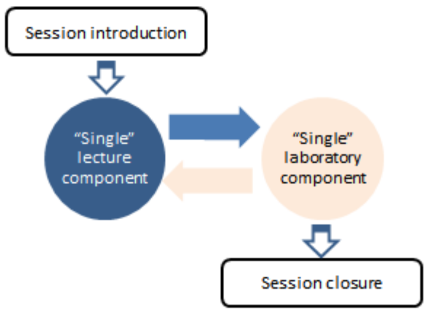

Figure 1. First design model for face-to-face teaching sessions for computing courses extensively using enterprise level systems in their curriculum

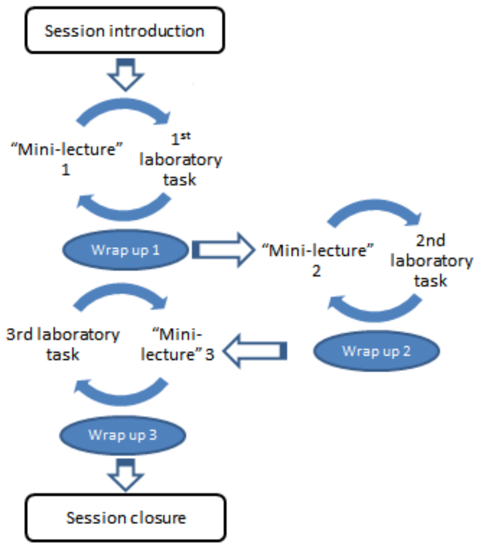

Figure 2. Second design model for face-to-face teaching sessions for computing courses extensively using enterprise level systems in their curriculum

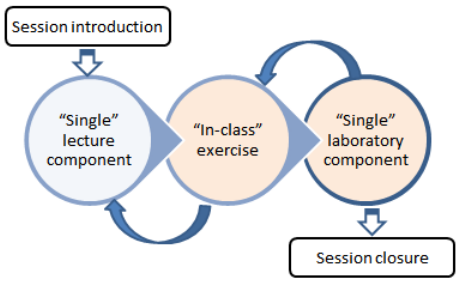

Figure 3. Third design model for face-to-face teaching sessions for computing courses extensively using enterprise level systems in their curriculum

portal topology basically by using "pen and paper" approach and, secondly, turning this conceptual design into a real implementation through setting up and configuring all the applications, site collections, sites and sub-sites required for the particular enterprise portal implementation. Thus, when using the third model, the students will be principally following this flow of tasks – firstly, they will be introduced to the concepts of portal topology, sites, applications etc. through the lecture component, secondly, they will design the portal topology for a given case study during the in-class exercise, and, finally, they will turn this design into a real implementation during the laboratory component.

Considering the complexity of the concept which needs to be explained (in this example portal topology) the use of an "intermediary step" (in this example in-class exercise asking students to reflect on a case study and draw a portal topology design) represents an effective way of linking the theory covered in the lecture component to the final laboratory component (in this example practical implementation of the applications, site collections and sites). It would be far more difficult (if not impossible at all) to make all participating students to grasp this linkage when completely removing the in-class exercise at all and trying to move from the lecture part directly into the practical implementation. As observed by the author of the paper, in such cases, only a very small fraction of students are able to appreciate the value of the laboratory exercises and understand what those laboratory exercises are attempting to illustrate and to showcase.

VI. SUMMARY AND CONCLUSION

In this paper, the author has discussed selected challenges and difficulties which arise from the use of complex enterprise systems and applications in university-level computing education courses. Contrary to professional or vocational courses, university courses are expected not only to equip students with practical skills and abilities in using, implementing and supporting such enterprise-level systems but also to enable students' understanding of organisational impact of such large enterprise-level system implementations.

While there are certainly different approaches possible to design face-to-face sessions for such courses, some of the suggested ways of proceeding represent the result of year-long experimentation and might be of interest to many an instructor attempting to integrate enterprise-level information systems into university curriculum. The most important aspect of the presented set of best practices is the attempt to enable the clear linkage of high-level concepts taught in the course to the practical skills acquired in the course. If this clear linkage is missing or not fully understood by the students, the academic nature of the course is potentially endangered and the course might be perceived as a unit imparting pure technical skills instead of offering broader, deeper and more profound view to the implementation of enterprise-level systems and applications.

REFERENCES

[1] Peslak, A.R., Twelve-step, multiple course approach to teaching enterprise resource planning. Journal of Information Systems Education, 2005. 16(2): p. 147-156.

[2] Targowski, A., & Tarn, J. M. , Enterprise Systems Education in the 21st Century. . 2007: Hershey, PA: Information Science Publishing.

[3] Hawking, P., Teaching Enterprise Systems Curriculum in Developing Countries. . International Journal of Learning, 2011. 18(23): p. 367-376.

[4] Hawking, P., McCarthy, B., & Stein, A., Second Wave ERP Education. Journal of Information Systems Education, 2004. 15(3): p. 327-332.

[5] Hawking, P., Shackleton, P., & Ramp, A., IS' 97 Model Curriculum and Enterprise Resource Planning Systems. Business Process Management Journal 2001. 7(3): p. 225-233. http://dx.doi.org/10.1108/14637150110392700

[6] Wang, M., & Hwang, D. , An Innovative Framework of Integrating ERP into IS 2010 Model Curriculum. Communications of the IIMA, 2011. 11(3): p. 75-86.

[7] Leyh, C., Teaching ERP systems: Results of a survey at research-oriented universities and universities of applied sciences in Germany. Journal of Information Systems Education, 2012. 23(2): p. 217-227.

[8] Rosemann, M., & Maurizio, A. A. , SAP-related Education -- Status Quo and Experiences. Journal of Information Systems Education, 2005. 16 (4): p. 437-453.

[9] Cameron, B.H. Enterprise Systems Education: New Directions & Challenges for the Future. in AMCIS 2008 Proceedings. 2008.

[10] Atif, Y., Al-Jaroodi, J., Alkobaisi, S., Jaffar, A., Ditsa, G., & Campbell, P., Enterprise Systems: Curriculum design and assessment. Education and Information Technologies, 2011. 16(4): p. 441-461. . http://dx.doi.org/10.1007/s10639-010-9138-4

[11] Swanson, Z., & Hepner, M. , Knowledge Management ERP Curriculum Design/Mapping (Theory and Development Tools). Decision Sciences Journal of Innovative Education, 2011. 9(2): p. 209-226. http://dx.doi.org/10.1111/j.1540-4609.2011.00304.x

[12] Wagner, W.P., & Pant, V. , Using Virtual Servers to Teach the Implementation of Enterprise-level DBMSs: A Teaching Note. Journal of Information Systems Education, 2010. 21(4): p. 349-354.

[13] Davis, C.H., & Comeau, J., Enterprise Integration in Business Education: Design and Outcomes of a Capstone ERP-based Undergraduate e-Business Management Course. Journal of Information Systems Education, 2004. 15 (3): p. 287-299.

[14] Seethamraju, R., Enterprise systems (ES) software in business school curriculum: an evaluation. Journal of Information Systems Education, 2007. 18.(1): p. 69-84.

[15] Fedorowicz, J., Gelinas, U. J., Usoff, C., & Hachey, G. , Twelve tips for successfully integrating enterprise systems across the curriculum. Journal of Information Systems Education and Information Technologies, 2004. 15(3): p. 235-244.

[16] Desai, M.S., & Pitre, R. , Developing a curriculum for an on-line international business degree: an integrated approach using systems and ERP concepts. Education, 2009. 130(2): p. 184-194.

AUTHOR

Ilse Baumgartner is an Adjunct with the School of Information Systems, Singapore Management University (e-mail: ibaumgartner@smu.edu.sg).

This article is an extended and modified version of a paper presented at the EDUCON2014 conference held at the Military Museum and Cultural Center, Harbiye, Istanbul, Turkey, 3-5 April 2014. Submitted, 03 July 2014. Published as resubmitted by the author on 05 October 2014..