Implementing an Integrated Curriculum with an Iterative Process to Support a Capstone Course in Information Systems

Bryan Reinicke
reinickeb@uncw.edu

Thomas Janicki
janickit@uncw.edu

Judith Gebauer
gebauerj@uncw.edu

University of North Carolina Wilmington
Information Systems and Operations Management
Wilmington, NC 28403

Abstract

Learning is enhanced with repetition, either through more exercises in individual courses, or through the integration of concepts in a capstone experience. A well planned and integrated curriculum can utilize a capstone course, not only to provide a service learning component, but also as an opportunity to refresh students on key discipline topics immediately preceding graduation. This article describes the process used at one university to integrate concepts taught in pre-requisite courses into the capstone experience. In addition, it discusses the need to constantly refine all of the courses to integrate the concepts and learning experiences in both directions. The capstone course must provide repetition and hands-on learning of earlier concepts, and the pre-requisite courses must provide the knowledge to enable a successful capstone experience for students. This is a two way integration up and down the chain of courses and instructors must work together to integrate all of the courses in the discipline to enrich the capstone experience and achieve desired learning objectives.

Keywords: Capstone Courses, Curriculum Development, Integrated Curriculum, Service Learning

1. INTRODUCTION

Many schools offer capstone courses as the final requirement in their respective disciplines. The learning goals behind a capstone course are many and vary based on the discipline. They may include: a) to integrate topics from various classes in one discipline, b) to bring rhetorical concepts from previous classes into the real world via a project, and c) to refresh the students on core principles in their discipline before graduation. Regardless of the specific goals, the instructor of the capstone course may be faced with a significant challenge if the pre-requisite courses to the capstone are done in a vacuum and the integrated nature of the capstone is not considered throughout curriculum development.

Tappert and Stix (2012) relate that the goal of a capstone is to familiarize students with how their trade is plied in organizations, so that the curriculum delivers the “practice" part of the promised "theory and practice."

The purpose of this paper is to explore how Information Systems (IS) curriculums can be designed within a single program to support a capstone experience. To do this, the authors will examine the literature in the area and reflect on over five years of experience at their host school in attempting to make the capstone course a positive learning experience for the students.

2. LITERATURE REVIEW

Importance of a Capstone Experience

Over a decade ago, Gupta and Wachter (1998) supported the need for a capstone IS course to “develop student abilities and skills needed in the integrative information systems technology and business areas.” Clear, Goldweber, Young, Leidig and Scott (2001) also encouraged the need for a project (capstone) under supervision where students apply what they have learned in their program of study.

Since that time, Abrahams (2010), Umapathy and Wallace (2010), Shih, LeClair and Carden (2010), Stillman and Peslak (2009) and Hashemi and Kellersberger (2009) have all discussed the importance and delivery of capstone or project-driven courses at their institutions. Capstone projects are widely used in business degree programs (Payne, Flynn, and Whitfield, 2008) to provide students with an opportunity to work on a ‘real life’ project.

The importance of capstone courses may be found in many of the accrediting agencies for colleges and universities. The Southern Association of Colleges and Schools (SACS) determined that an integral part of their Quality Enhancement Program (QEP) is the consideration of capstone experiences, defined as a senior level course that empowers students to evaluate, appreciate and integrate multiple perspectives in a collaborative project (2012 website).

The 2010 Model IS Curriculum (Heikki et al., 2010) lacks a capstone project course in the core model curriculum. However, Schwieger and Surendran (2011) argue for the need to incorporate a capstone course in the IS 2010 Model Curriculum.

Learning Theory

Constructivist learning theory emphasizes the usefulness of combining and building on previous knowledge that typically happens in a capstone course. For example, Brandt (1997) states that learners construct new knowledge by making sense of experiences in terms of what is already known. Learners transfer knowledge through experiences via mental models, which are used to assimilate new information into knowledge, and thus become expanded mental models. This knowledge transfer emphasizes knowledge construction and problem solving in domains.

Rakes (1996) recommends increasing students’ success through the addition of practice and through a shift from the traditional theories of learning (cognitive and behavioral) to a resource-based view of learning. The resource-based view of learning involves the role of an instructor changing from an expert dispensing knowledge to a “guide” providing resources, and requires an increase in the number of problems, assignments, and exercises given to the students (Rakes, 1996). Finally, Yadin and Or-Bach (2010) discuss the continuing need for self-assessment and multiple individual exercises in an environment of collaborative learning.

Capstone experiences fit these learning theories well and support the construction of new knowledge through the integration of concepts from prior courses. In addition, the capstone project enables students to have individualized (or team) projects that enable them to self-explore, which is in line with the resource-based view of learning.

Service Learning

Many capstone courses include a service learning component as well. Preiser-Houy and Navarrete (2012) discuss service learning in the context of a teaching strategy that integrates discipline-based learning with relevant community service.

Wei, Siow and Burley (2007) discuss the fact that service-learning is an educational strategy that combines classroom-learning experience with a community service experience. It

requires that courses be modified to involve real projects from communities and thus provide students with real-world experiences in a relatively safe academic environment (Wilcox and Zigurs 2003). Service learning has multiple benefits for students by engaging them in the community in which they are learning and allowing them to develop leadership skills as they work through the project (Rose et al. 2005).

3. THE CAPSTONE EXPERIENCE

The IS major at the authors' university is a stand-alone program within the School of Business with approximately 100 majors out of roughly 2,000 undergraduate students. The program has included a capstone experience as a part of the curriculum for approximately eight years. Over the last five years, the faculty has been focusing on continually improving the learning experience, with an emphasis on how to better integrate the learning concepts and theories from the other IS courses so that the capstone course could build (constructivism) on these theories to truly enhance student learning and retention of this knowledge.

The capstone experience is comprised of the following key components:

- Real-world IS development projects with external clients
- Teams of two students work on individual projects
- n-Tier system development environment
- Each project has a back-office SQL database
- Each project generally has a web portal to support the desired functionality of the client
- Teams must develop a presentation layer
- Teams must develop a business logic layer

The following courses are considered pre-requisites for the capstone, as they introduce students to various concepts that are then reinforced during the capstone course:

- Database Management (SQL)
- Business Software Development (VB.Net)
- Systems Analysis and Design (SAD)

The SAD course in particular has become an integral part of the capstone, as this is the course where students meet with the end-clients, develop system specifications to meet client needs, and prototype potential solutions. A contract is signed with the-end client in this course.

As a part of integrating the SAD course with the capstone, many changes needed to be made to the SAD course. These changes necessitated altering the syllabus to include a number of deliverables for the capstone project. One of the problems that had emerged in the capstone course was that the students did not have sufficient time to gather and document requirements and build the system in a single semester. Thus, the tasks for gathering and documenting requirements were moved into the SAD course. In all, the deliverables for the capstone project count for 35% of the SAD course grade. One other change was to move the SAD course to an object-oriented methodology. This more closely matched what was being done in the development courses, as well as reflecting current industry trends towards object-oriented development.

As one of the deliverables in the SAD course, students must develop an interview guide and meeting agenda prior to meeting with the client. This exercise forces the students to think about what information they must gather from the clients ahead of time, and gives them the experience of planning a meeting with a client. After this deliverable has been graded, the students must schedule a meeting with the client and document the meeting as a deliverable for the class. Once the students have met with the client, they have to develop the scope document that was mentioned earlier. They are required to present this scope document to the client in order to get the client's sign-off on the plan for the project. This is a good practice for "real world" projects, and forces the students to think about what the project should actually entail.

The next deliverable is a prototype for the system. While the students do not need to produce a working prototype (this is an exercise for the capstone course), they do need to model every screen that will be included in the system. After a number of iterations, the instructor for the SAD course moved this exercise to earlier in the semester, as creating a prototype forces the students to think about the system as a whole – something they have never had to do before.

The students are then required to develop a set of use case diagrams. This exercise also forces the students to think about the system in detail (which users will perform what functions?), and often prompts changes to the prototypes. This also introduces students to the concept that

most projects are iterative in nature. SAD courses tend to emphasize that development is an iterative process, but most students have never actually experienced this. Generally speaking, the students are stunned to find out that: a) their first deliverables were not perfect and b) if they don't correct the earlier deliverables, they will be unable to complete the project next semester.

At this point, the students are required to meet with their clients to present the (now revised) prototype for feedback. The students are again required to plan the meeting and submit meeting notes as deliverables for the class. This mid-point check was included to make sure that the students did not spend the semester working on a system that did not meet the client's needs. Of course, depending on the client's feedback, the students may (again) be required to modify their prototype to meet the client's needs. This is also generally the point at which the students gain their first experience with scope creep, as clients will frequently try to add additional requirements to the system.

Next, the students have to develop descriptive use cases, which forces the students to think about the business logic and information flows for the system. The authors have noted that this assignment is frequently a problem area for students, as they have rarely had to develop their own logic in the past, and tend to be challenged when having to consider "what now" questions. How should the system respond to input? The new understanding can lead to more changes to the prototype, when it triggers the realization that important functions have been left off of one or several of the screens.

The students then must develop an Entity Relationship Diagram (ERD) for the database design. Again, this is frequently challenging for the groups as they have, again, never really had to develop an ERD to support a system. The groups generally discover that a working system requires many more tables than they anticipated. It is not unusual for this activity to prompt the groups to refine their prototype again, as they discover that to make the system work they will need to add in additional detail, which requires that detail to be entered on a screen within the system.

In order to link the use cases and ERD, the students must develop a Create, Read, Update, Delete (CRUD) diagram. Once again, this exercise proves very useful to enhance learning and deepen understanding. Students are often quick to indicate that they fully understand CRUD diagrams following their introduction in lecture, only to find out that actually creating one for a system is significantly more difficult than they had anticipated. This exercise is also useful for forcing the students to think about how the system should actually work, and eventually triggers "aha"-effects when students realize the inherent connection between the ERD and use cases that are conceptualized in the CRUD diagrams.

In order to expose the students to the concepts of project management, and begin to map out the various activities in the capstone course, the students are required to develop a project management plan. This exercise requires the students to map out what has been done in the SAD course and to begin mapping what they think needs to be done in the capstone course. Again, this is generally their first exposure to these concepts, and they discover quickly that developing a detailed project management plan requires a great deal of attention to detail.

Near the end of the SAD course, the students are required to check in with the client one more time at the end of the semester to present their (now extensively refined) prototype to verify the direction the project is taking. The students are again required to document this meeting as a deliverable.

Finally, the students must present their projects to the class at the end of the semester. There are generally multiple projects each semester, so each group could be working on a separate project. The students have to present and explain their prototypes, and describe how they envision the system working once it is completed. The end-of-semester presentations often result in lively discussions and questions from the class on their fellow students' projects, which appears to demonstrate the deep insights that students have gained about the SAD process.

One thing should be noted for the instructors of SAD courses who are considering the approach just described. Asking the students to make multiple changes to their deliverables also requires re-grading the deliverables multiple times. If the students do not make corrections as they go along, they will not have the designs they need going into the capstone course. Thus,

to encourage this additional work on their part, the deliverables must generally be re-graded several times. However, in order to encourage good work the first time around, it is a good practice to set a maximum increase for resubmissions. This limit should be high enough to make it worth the students while, but not so high that it diminishes the effort prior to the first submission.

As can be seen from this discussion, the integration of the SAD course with the capstone course has occurred at a significant level, and students appreciate that they are working toward the same end-project in two subsequent courses. The faculty have noted that projects have become more robust as a result of having two semesters to work on the same end-client project. One remaining challenge is to better integrate the database and the business application development courses into the learning outcomes that will be needed in the capstone and SAD courses.

Finally, an additional motivation to integrate the curriculum even more is the fact that the authors' host university is modifying its graduation expectations to include more integration, more service to the community and more experiential learning.

4. LESSONS LEARNED

Adjustments must be made

Developing an integrated curriculum is not a one-off activity. It is a process that requires an annual review by the faculty who are impacted by it. To assist in the course integration, the faculty involved in the four classes mentioned earlier meet periodically to discuss how the courses could be better integrated.

This annual review by faculty also takes into consideration written reviews from students and suggestions from the end-clients. Added to these formal reviews are the faculty's overall evaluations of student projects in both the SAD and capstone courses.

Summary of challenges from annual reviews

Challenges that appeared in both the SAD and capstone courses include: the move from theory and conceptualization (database, development, and SAD courses) to implementation (capstone course) is more difficult for students than they first expect. It is one thing to design a few tables and relationships in a database course, and another one entirely to be in a "production" environment which requires the building of 50 plus tables with the corresponding relationships. Another challenge that students encounter is moving from a logical ERD to a physical database design.

In the SAD arena, students have difficulty when they try to go from use case descriptions and CRUD diagrams in SAD to actually building the proper stored procedures, functions and objects in the capstone course. In addition, a common refrain heard from the students is that they didn't receive all the information needed from the client. Or that the client didn't tell them what they really wanted. Of course, this is a common refrain heard from systems developers as well, which simply emphasizes why this type of learning experience can be so valuable for the students.

From the software development side, the enhanced use of business objects, business logic layers and building the logical flow for the menus, actions and reports can be overwhelming for new developers where the exact steps have not been detailed for them in advance.

When it comes to the replication of real-life experiences, students also get exposed to the critical need for a disciplined approach to managing their projects. As laid out earlier, the SAD course, as well as the subsequent capstone course, requires the students to develop, submit, and possibly revise and resubmit, a sizeable number of deliverables as a team, and also in collaboration with the end-client. While students continuously work on their project deliverables, new concepts and skills are being taught and practiced in the classroom, with the expectation that these will be applied to the project shortly thereafter. So, time management is of critical importance and procrastination can have serious effects on the success of the projects, as a result of the close link between concepts and application in both courses.

An added hurdle results from the fact that students effectively work with two 'bosses' (faculty members) as they move from one course (SAD) to the other (capstone). Deliverables need to be handed over, and

possibly revised again once students start working on their capstone projects, which even though not necessarily ideal from the viewpoint of the students is certainly not unusual in real-life projects.

Can you change your syllabus?

When attempting to develop an integrated curriculum, one of the key problems is that what a student learns in one course should be applied in another. This means that all of the faculty involved must make occasional adjustments to their syllabi and assignments to better meet this goal.

The process of developing an integrated curriculum requires discussion among the faculty, and will likely require changing the syllabi for other courses as well. Such changes should not be viewed as an attack by other faculty, but rather an acknowledgement that there is a chance for their course to better support the capstone. Still, some faculty may be inclined not to change their syllabus and learning concepts. The authors have noted that student comments on their previous learning experience (gathered at the end of the capstone course) can help to overcome this resistance. In these cases it can be important to stress that students know the concepts, but are having trouble bridging theory and practice.

One of the revelations of our integrated approach was that achieving buy-in from all faculty involved is not necessarily easy. This is a process, and must be approached as such. This isn't surprising, in retrospect, as it does require that faculty surrender some aspects of their curriculum to support what is needed for the capstone.

Change is the only constant

When using an Integrated Development Environment (IDE), whether it is Microsoft's Visual Studio or Eclipse, the only guarantee is that it will change within the next two years. This presents a special problem for integrated programs, because it is likely that the software will change between the time students take the intro programming course and the capstone. Thus, when moving to a new software platform, special care has to be paid to who is moving to the new platform and when. This can create problems for faculty who may be working with different versions of the same software, as well as for a University's technology department, as they may then have to support multiple versions of the IDE simultaneously.

Sharing work files between classes

While the students' (sub-optimal) habits with regard to saving and naming files may not seem like a problem for faculty, they become one when the curriculum becomes more integrated. If the deliverables from one course feed into another, those files need to be stored such that the students can readily access them. Somewhat surprisingly, file access has been a nearly constant headache, particularly for the instructor of the capstone course, as students tend to misplace and lose files for any number of reasons. Given the extent to which the courses and deliverables build upon one another, the loss of files can be disastrous.

One solution to this problem could be a robust class management system, such as Blackboard, as long as it is set up to allow students access to deliverables from a previous semester. Another solution could be a network or cloud-based drive that is backed up regularly where all of the group members have access to the files. The authors would recommend that such a solution not be hosted by the faculty concerned, but rather by the university or an outside entity (such as Google).

A group is only as strong as its weakest link

The suggested high level of integration also requires that all faculty members possess comparable practical skills which probably requires more frequent updates than a less integrated situation would. Ours is not a field that stands still, and if there are faculty members who are not committed to updating their skill sets on a regular basis it can be incredibly difficult for students to obtain the skills and knowledge needed to be successful in the later courses, and especially the capstone.

These observations are based on the authors' experience with developing both IS capstones and an integrated IS curriculum at their university. However, these are issues that any program will encounter when moving toward a more integrated curriculum.

Examples of changes made to the courses

Earlier, a list of student difficulties in transferring theory to practice were detailed. In this section, we summarize some specific changes made to four courses in the IS curriculum as a result of the integration.

Database Management: Originally, stored procedures were mentioned, but not stressed in the database management course. Now, a more significant portion of the class is dedicated to the value and building of stored procedures

Business Software Development: In the context of an integrated curriculum, the course has been changed to put more emphasis on the use of business objects (e.g., person, invoice, inventory item). Now, students must construct multiple objects and understand the purpose of inheritance and isolation. In addition, this course has been changed to include more database applications. This is done so the concepts learned in the Database Management course are reinforced earlier. Items of SQL connections, strings, and an entire CRUD process is developed and practiced.

Software Analysis and Design: A number of adjustments to this course have been described earlier in this paper. In addition, this course has also been changed to include the use of SQL software for ERDs and DFDs. Students now also build a small segment of their database for the end client. Use case descriptions and use case diagrams have been enhanced to emphasize the use of menus by role which helps students build the logical flow in the capstone courses. Taking the use case diagrams with the depiction of different actors and translating them to menus and menu items in a prototype system has really enhanced students' grasp of applying the concepts to reality.

Capstone Course: For a smooth transition between the prerequisites and the final capstone course, the instructor of the capstone needs to be aware and considerate of the concepts and skills introduced and practiced in the previous courses. Before beginning the work on the actual capstone projects, it has proven useful to reinforce previous learning and bring everyone up-to-date. The first month of the class has, thus, been modified to include assignments on building various stored procedures for a common table (i.e. valid states). Having the entire class work on a common assignment and build four important procedures (read, update, insert, delete) has made their development easier later in the semester. In addition, more time is now spent on use cases to help build the business logic layer, especially in the area of 'exceptions' noted in the use cases.

5. CONCLUSIONS

We believe the annual reviews of the involved faculty has increased the level of integration in the IS curriculum and has enhanced student learning. As a result of the integration of the courses, more of the projects reach the 100% completion rate, making the end clients much happier. In some respects the capstone course has become a great refresher on many IS skills needed by employers right before students graduate, as well as a valuable integrative educational experience. The project itself is just a vehicle to help translate theory to practice.

6. REFERENCES

Abrahams, A. (2010). Creating e-Commerce Start-ups with Information Systems Students: Lessons Learned from New Venture Successes and Failures. *Information Systems Education Journal*, 8 (35).

Brandt, S. (1997). Constructivism: Teaching for Understanding of the Internet *Communications of the ACM*, 40 (10), 112-117.

Clear, T., Goldweber, M., Young, F.H., Leidig, P., Scott, K. (2001) Resources for instructors of capstone courses in computing, *ACM SIGCSE Bulletin*, 33(4).

Gupta, J. N. D. Wachter, R. M. (1998). A capstone course in the information systems curriculum. *International Journal of Information Managemen*t, 18 (6), 427-441.

Hashemi, S., Kellersberger, G., (2009). The Pedagogy of Utilizing Lengthy and Multifaceted Projects in Capstone Experiences*. Information Systems Education Journa*l, 7 (17).

Heikki, T., Valacich, J., Wright, R., Kaiser, K., Nunamker, J., Sipior, J., deVreede, G., (2010). Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *Association for*

Computer Machinery (ACM) and Association for Information Systems (AIS).

Payne, M., Flynn, J., Whitfield, J.M., (2010). Capstone Business Course Assessment: Exploring Student Readiness Perspectives. *Journal of Education for Business*. 83(3).

Preiser-Houy, L., Navarrete, C. (2011). A Community-Based Research Approach to Develop an Educational Web Portal *Information Systems Education Journal*, 9(1) 4-13.

Rakes, G. (1996). "Using the Internet as a tool in resource based learning environment". *Educational Technology, 6* (2), 52-29.

Rose, D. Meyer, A., Hitchcock, C., (2005). The Universally Designed Classroom: Accessible Curriculum and Digital Technologies. *Harvard Education Press*, Cambridge.

Schwieger, D., Surendran, K. (2011). Incorporating Capstone Courses in Programs Based on the IS2010 Model Curriculum. *Information Systems Education Journal*, 9(2) 65-74

Shih, L., LeClair, J., Varden, S., (2010). The Integrated Technology Assessment: A Portfolio-based Capstone Experience. *Information Systems Education Journal*, 8 (63).

Stillman, R., Peslak, A., (2009). Teaching Software Engineering Including Integration with Other Disciplines. *Information Systems Education Journal,* 7 (40).

Tappert, C., Stix, A. (2012). Adapting to Change in a Masters-Level Real-World-Projects Capstone Course. *Journal of Information Systems Educators*, 10(6) 25-37

Umapathy, K., and Wallace, F.L. (2010). The Role of the Web Server in a Capstone Web Application Course. *Information Systems Education Journa*l, 8 (62)

Wei, K., Siow, J., Burley, D., (2007). Implementing Service-Learning to the Information Systems and Technology Management Program: A Study of an Undergraduate Capstone Course. *Journal of Information Systems Education*, 18(1), 125-136.

Wilcox, E., Zigurs, I., (2003). A Method for Enhancing the Success of Service-Learning Projects in Information Systems Curricula. *Information Systems Education Journal*, 1(17).

Yadin, A., Or-Bach, R., (2010). The Importance of Empahsizing Individual Learning in the "Collaborative Learning Era". *Journal of Information Systems Education*, 21(2).