ISSN: 2473-3857
v2 n4034

Performance Learning of Agile Methodology Using Paired Courses of Systems Analysis and Design and Web / Mobile Programming

Edward V. Weber
EWeber@Millikin.Edu
Tabor School of Business
Millikin University
Decatur, IL 62522, United States of America

Abstract

In an Application Development specialization within an Information Systems curriculum, the concepts of the Agile methodology of application development are often explored in a variety of courses including Systems Analysis and Design, Web and Mobile Application Design, and various levels of Programming courses such as Introductory, Intermediate, Advanced, Web, Mobile, etc. Too often, however, these Agile concepts are only being presented and discussed as a presentation of theory. Within each isolated course the Agile concepts typically cannot be fully practiced for one primary reason: there may not be sufficient resources available having the required complimentary skill sets to complete the Agile team environment. In other words, rarely can the Agile concepts be fully *practiced* within the confines of a single course. This paper seeks to inform how two courses have been successfully paired together to afford the Performance Learning aspect of implemented and practiced Agile development within both courses.

Keywords: paired courses, Performance Learning, IS Curriculum, Agile methodology, application development, analysis and design, programming, undergraduate education

1. BACKGROUND

IS Curriculum Content

Undergraduate students in an Information Systems program consume a variety of courses to develop a diverse set of skills which fully prepare them for their future roles within an IS organization. The Association of Computing Machinery (ACM) has created, updated, adopted, and otherwise provided curricula recommendations for years. The IS 2010 Curriculum Guidelines provides an excellent framework for structuring an overall IS program and is used by many institutions in the planning and development of their IS curricula. (Topi, Valacich, Wright, Kaiser, Nunamaker Jr, Sipior, & de Vreede 2010) While there does seem to be debate in the literature as to how much the IS 2010 Curriculum Guidelines are being implemented or to what degree they are being strictly followed, much of this debate seems to stem around the differentiation between the most fundamental of program definitions (e.g. between IS, vs. IT, vs. CIS, vs. MIS, etc.) For the remainder of this paper, it should be noted that this approach has been endeavored within an Information Systems program which is situated within the Tabor School of Business at Millikin University, a small private University in Decatur, Illinois.

When considering material for inclusion within specific IS courses, these guidelines, coupled with professional experiences may help to inform a content developer regarding specific topics and methods of delivery. For example, it is a given

that any introductory Programming course will necessarily include concepts such as variables, data types, expressions, operators and operations, decision logic, looping, etc. Likewise, a Data and Information Management course will include concepts such as data models, entity relationships, object-oriented models, data types, indexing, the role of a DBMS, database languages like SQL, DDL etc.
In these basic descriptions defined above, the observant reader may have recognized the intentional duplication of the topic named 'data types'. This is to help the reader recognize a typical pedagogical concept of topic overlap between courses.

Content Overlap
One ongoing concern for course developers and deliverers is how to ensure that sufficient resources are dedicated to each critical topic in each course while acknowledging that certain topics will necessarily be repeated, in varying degrees (either as original material or as review material), across multiple courses. For example, the concept of 'data types' will necessarily be discussed in each of the Programming courses as well as in the Data and Information Management courses as well as in the Analysis and Design courses. Considering that these individual courses may not typically be required to be taken in series, it becomes obvious that within each course, certain overlapping concepts may become an issue of either under- or over-coverage and this variability occurs between courses and between students within courses.

Course content developers and deliverers can often 'discover' which overlapping topics need additional coverage for their students and which have been sufficiently mastered and can typically adjust their lesson plans accordingly. In other words, if a basic review or pre-test of the concept of 'data types' reveals insufficient mastery of the concept, the required material can be covered and appropriate practices can be undertaken and assessments can be made to insure sufficient concept mastery.

One of the biggest problems with this approach is that from course to course and even from student to student, there is an ongoing risk that there will be a significant gap or range between students with insufficient topic mastery and those students with topic proficiency.

As a result, resources external of the course meeting times can be made available to help bridge this gap. Everything from asynchronous online materials, practice materials, and one-on-one tutoring are just some of the options for this type of remediation. For example, after the successful completion of one or two courses that included the concept of 'data types', a course deliverer will have an expectation that the students have a sufficient mastery of this concept. If a student appears to be lacking at this time, these aforementioned remedies can be employed to bring this student's performance up to the required level to enable the student to then continue with the new content. Therefore, it is safe to say that while course content overlap is not a new discovery nor are its methods of remediation new, it does, in fact, still remain a pedagogical issue.

Curriculum Model Evolution
With the IS 2010 Curriculum Guidelines, there was an intentional 'flattening' of the curriculum structure from the previous IS 2002 model in a direct effort to "…offer a flexible structure that can integrate electives easily" (Topi et al., 2010). This flattening has resulted in the removal of intentional sequencing of courses. This has resulted in a necessary increase in the number of topics that experience this type of problematic content overlap as previously discussed.

While some of this additional content overlap can be absorbed into the resulting new curriculum via the previously defined remedies, there is at least one topic (and most probably more) that cannot be sufficiently addressed within the constructs of a single course.

2. PROBLEM STATEMENT

In the previously discussed examples, specific course content and concepts may necessarily overlap among multiple courses. As a result, at any of these various consumption points within the curriculum, individualized techniques may be utilized to ensure that the students achieve sufficient skills mastery. However, certain new and evolving concepts and content cannot be managed in this traditional way and require additional or different methods of ensuring sufficient student skills mastery.

One such topic is the Agile methodology and how it contrasts significantly from the traditional Waterfall methodology for Systems and Application Analysis and Design.

In the traditional Waterfall methodology of the Systems Development Life Cycle (SDLC), a

ISSN: 2473-3857
v2 n4034

definitive sequence exists (Figure 1) which allowed for curriculum development to mirror the sequence of critical skills acquisition.

For example, the critical skills necessary to execute the Development phase of the Waterfall SDLC methodology could be fairly isolated within the learning objectives associated in the Programming courses with only minor overlap between the Design and Testing phases. Likewise, the critical skills necessary to execute the Requirements and Design phases could be fairly isolated within the learning objectives of the Analysis and Design courses with only minor overlap with the Development phase.

Figure 1 - Traditional Waterfall SDLC

However, just as the IS 2010 Curriculum Guidelines now recommends the flattening of the curriculum to intentionally remove the sequential nature of content presentation, this flattening is likewise represented in the newer application development methodologies such as those represented in the Agile methodology (Figure 2.)

Whereas the Waterfall methodology represents a single flow of activity over the entire span of the project, the Agile methodology represents an iterative collection of activities that are, themselves, repeated multiple times in short bursts or cycles.

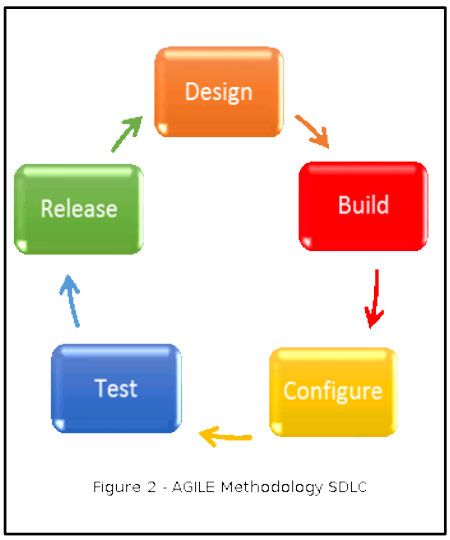

Figure 2 - AGILE Methodology SDLC

The Agile Manifesto states as one of its core principles an intent to "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale" (Fowler & Highstreet, 2001). As a result of this cyclic nature, the entire complement of skillsets required to effectively operate in an Agile environment must be fully developed or available.

With the Agile methodology, application development exists along a continuum of work with skillsets consistently overlapping. The cyclic design of this methodology requires ongoing and continuous overlap of skillsets and is typically implemented using team members with expertise in the multitude of skill areas.

While the accumulation of some of the required skills necessary to successfully operate within an Agile methodology can still be fairly isolated within specific coursework, certain aspects of Agile methodology requires the existence of a multitude of skills which often are not fully developed within a single course.

One approach to teaching Agile and some of the subordinate components of Agile such as Scrums, XP, etc. is to utilize a full course as Mahnic presents by allocating a capstone course specifically on Agile Software Development using Scrum. (Mahnic, 2012) In this approach, Mahnic states, "In pursuing the challenge of combining formal with practical learning, the design of the capstone course can rely on previous courses having provided formal lessons on the aforementioned topics, so just a short theoretical introduction to agile methods (including the reason for their importance) is

ISSN: 2473-3857
v2 n4034

enough to encourage students' buy-in before starting the practical work." (Mahnic, 2012).

Considering the significant growth of Agile in the IS workplace, it seems more desirable to have earlier incorporation of this methodology in multiple places within the undergraduate curriculum. This also affords the student multiple opportunities for Performance Learning of this critical component rather than waiting for a single capstone course to serve as a single source for this experience.

But unlike Mahnic's capstone course approach, the paired class approach understands that not all of the individual students will have all of the requisite skills for full participation in the Agile team.

For example, Cervone defines an Agile Scrum team as a cross-functional team of five to ten people who work on the project full time (Cervone, 2010). Within a cross-functional Agile team, the skills of needs assessment, analysis, design, programming, testing, and documentation are all required. Typically, these skills are found in the various team members including business unit experts, analysts, developers and other members of a fully cross-functional team.

But all of these skills are not fully developed for undergraduate students within a single course content. Therefore, in traditional undergraduate courses, some required content is typically sacrificed and, as a result, the practice of Agile concepts cannot be fully representative of how the work is accomplished in 'the real world.'

Consider attempting to practice the tenets of Agile methodology within programming courses. For programmers to be productive, the required definitions of application inputs and system outputs must be available and these are typically identified within the context of analysis and design. Therefore, although these components are required in order to complement the programming learning, these components must be artificially or externally supplied if the programming course was not sequenced within the curriculum to reside after an analysis and design course. In other words, programming students who have not mastered the skills of analysis and design will be ill-prepared to fully produce without some external resources. And yet, the nature of the Agile methodology requires these skills to be available to the Agile team through its membership.

Whereas this sequencing of coursework was formally a part of the curriculum structure as well as a part of the Waterfall SDLC methodology, this sequencing is no longer supported in either recommended curriculum design nor in the Agile SDLC methodology. Therefore, the skills required to effectively practice the Agile methodology are not typically nor consistently available to all application development students at predicable intervals within the curriculum.

3. APPROACH

Performance Learning in IS

In order to provide a more accurate representation and direct opportunities to practice concepts learned in the classroom, Performance Learning is utilized within the IS curriculum at Millikin University in Decatur, Illinois. Performance Learning is defined at this institution as "…the opportunity for students to experience real risk and reward while having their work evaluated by a third-party stakeholder" (Podeschi, 2015).

In the Performance Learning environment of IS, course content is immediately applied by students utilizing newly acquired skills while working on real-world projects for real-world third-party stakeholders with real-world risk and rewards.

Performance Learning is typically not utilized during the foundational coursework. Likewise, in accordance with the IS 2010 Curriculum Guidelines, the foundational coursework is prerequisite to the more advanced courses within the Application Development curriculum.

However, after students have successfully completed Foundations of Information System and the other core IS courses, students may enroll in the Systems Analysis and Design course, or the Web/Mobile Programming course, or they may co-enroll in both in the same semester.

In both the Systems Analysis and Design course as well as the Web/Mobile Programming course, the concept of Agile development is fully explored and the relationship of these two courses becomes obvious. But in order to conduct Performance Learning with the concept of utilizing the Agile methodology, a single project needed to be chosen that would afford the opportunity for these two classes to work

together while independently learning the content that is unique to each course.
This technique has been successfully applied in the Fall semesters of 2014 and 2015. In the first year, an internal University project was undertaken to develop a fully automated, data-driven, web-enabled, self-maintaining, searchable, organizational hierarchy chart for the University.

In the second year, an external project was undertaken to provide consulting and systems enhancements to a home-grown inventory management, point-of-sale, and customer management web-based system for a local health food and supplement retail store.

Performance Learning of Agile
The implementation of Performance Learning for the concept of Agile methodology was achieved by pairing the two independent courses of Systems Analysis and Design and Web/Mobile Programming together to create a setting to more fully explore the Agile concepts within a single semester.

In each course, the first quarter was dedicated to new content and material that was specific to each course. Also in the first quarter, Agile methodology was explored in both courses with shared content, terminology, and discussions of how the concepts integrate within the other courses.

In the first year, the Systems Analysis and Design course had 16 students and the Web/Mobile Programming course had 7 students. In the first year, 3 of the students were co-enrolled in both courses.

In the second year, the Systems Analysis and Design course had 14 students and the Web/Mobile Programming course had 6 students. In the second year, 5 of the students were co-enrolled in both courses.

In both years, the Faculty member served as the Project Owner in conjunction with the third-party stakeholders. The primary reason for this was that the third-party stakeholders themselves were neither fully prepared to serve the role as content experts nor were they fully versed in the aspects of Agile development. This activity was, in fact, a learning opportunity for them as well.
In both years, the Agile methodology was utilized with narrowly specified objectives to accommodate the significantly reduced time frames within a single semester. The goal was to get the students into the Agile mindset of rapid development and delivery in an iterative mode.
In the first year, in precisely one class period, the Systems Analysis and Design class created a mockup layout of their desired design for the application and had documentation ready to present to the Web/Mobile class. In the following Web/Mobile programming class, the students created the prototype of the entire driver page which would become the backbone of the entire application. In just two 75-minute sessions, the framework of an entire application was analyzed, designed, built, and ready for initial testing by developers and users alike.

In the second year, the Systems Analysis and Design class had planned an initial scope of making desired enhancements to the existing client Web application. While a team from this class was meeting with the client to confirm project scope, the Web/Mobile Programming class created a development environment to mirror the existing client production environment. Both teams experienced significant requirements changes as a result of their efforts.

The Systems Analysis and Design class discovered that the author of the existing application had recently passed away and he represented the only IS personnel for the entire organization. Additionally, they learned that the author was self-taught, there was no system documentation of any kind, and there were some compiled C++ modules, in addition to the PHP web application, that were being used in production and which had no discoverable source code.

The Web/Mobile Programming class discovered that the existing web application was configured in a manner in which all critical errors, warnings, and deprecated code errors were being suppressed from displaying in the production environment. When they created a working development environment and configured it so that they could properly code, test, and debug, they discovered over 140 PHP, JavaScript, and HTML errors throughout the application. In addition, they discovered some non-normalized data structures in the existing database as well as some queries and views that were failing because they were running so long that they timed-out before completion.

When the two classes conducted their backlog meetings to detail their discoveries, they each

immediately experienced the benefits of Performance Learning in the Agile development environment. Both classes recognized that adaptation was necessary to reconcile this real-world client's actual needs vs. their perceived needs. Both classes adjusted the overall project objectives and deliverables with a new focus of creating a stable environment and documenting the existing systems to create a foundation for further analysis, design, and development. Performance Learning using Agile methodologies allowed the students to experience rapid development and the ability to adjust to rapidly changing requirements while remaining productive.

4. DISCUSSION

There are a number of successes and challenges to this approach of pairing two courses (Systems Analysis and Design and Web/Mobile Programming) for the intent of creating a Performance Learning environment for Agile development. First we look at the successes.

Successes

With the paired course approach, students from one class could be intentionally teamed with members of the other class and also have at least one member who was co-enrolled in both classes. This commingling of students across classes is, by far, the greatest success of this approach. By structuring the teams of the classes this way, the students immediately experience the cross-functional nature of Agile teams in that some students will have expertise (or personal preference) for the Analysis and Design work while others will be specialized in the Programming or technical aspects of the work. This allows the content deliverer to focus on the role of being the Team Leader and facilitator of the students' interactions with the client.

Another achievement of this approach is that students learn through Performance Learning that Agile succeeds through rapid performance in short, potent, iterative sprints. By intentionally narrowing the scope of the sprints, the students quickly experience the successes of their efforts which keeps them highly motivated.

Students who were co-enrolled in both classes consistently acknowledged how their co-enrollment afforded them a significant advantage in understanding the bigger picture of the project and also how it prepared them to recognize team member strengths and limitations and how to adjust their own efforts accordingly.

Delivering the course content of Agile concepts became easier because co-enrolled students became de facto teacher assistants or tutors of the concepts across both courses. Shared understanding became essential for both courses and was rapidly achieved.

Without this approach, Performance Learning of this concept cannot accurately occur in that there are insufficient resources in each of the single courses to properly satisfy the roles of the cross-functional team members. Without this approach, the content deliverer must attempt to satisfy all of the missing roles which does not provide an accurate representation of the actualities of Agile development.

Challenges

The biggest challenge that exists with this approach is the issue of time. Once students understand the fundamental concepts of Agile methodology, they quickly understand that the successes are achieved from the rapid cycles of Design, Build, Configure, Test, and Release. But they also recognize that having team members from both classes (as well as internal or external third-party clients) means that they will need to find time (outside of each of the regular class meeting times) to hold team meetings. This challenge is becoming one of the greatest issues across many undergraduate programs: Students are expected to work and perform in cross-functional teams with diverse membership but there are no intentional scheduled time allowances to afford the students time to meet.

In contrast, in the 'real world', most Agile team members are in the same organization and, as designated members of an Agile team, they are afforded intentional time allowances from their respective managers to perform the work required of an Agile team. In this paired course approach for Performance Learning of the Agile concepts, the content is more successfully delivered and most of the concepts become quickly absorbed by the students. But then, the reality sinks in. The multitude of students having significantly divergent courses and activities with often over-filled time demands creates an exceptional hardship in trying to get the students to find *coordinated time* for all of the critical Agile team member activities. This issue does not appear to be an issue of student motivation. In most cases, the students fully understood what was required and were willing

to fulfil their respective roles. However, there was consistent feedback that indicated that finding *coordinated time* amongst Agile team members (including both classes and the user community) was the number one impediment to the teams' ongoing productivity.

Another challenge to this approach involved the students who were co-enrolled in both classes. While some of these students expressed pleasure in the benefits of co-enrollment (i.e. seeing the 'big picture', serving as an intermediary, possibly taking leadership roles, etc.) others expressed frustration and feelings of being overwhelmed (i.e. feeling like they were doing most of the work because they were in both classes.) This challenge puts an additional burden on the deliverer of the course content to properly establish and communicate course and assignment rubrics that equitably and consistently describe individualized performance expectations.

Another challenge that exists in this approach is that by the nature of Agile development, once the concepts are mastered and the actual iterative cycles commence, there is a natural ramp-up that occurs and, very quickly, the whole process is moving ahead at a fairly fast pace. Students that are struggling with individual course content run the risk of quickly falling behind, and therefore may become underperformers within the Agile team, which becomes a risk to both their own motivation and success as well as the team's.

Course deliverers must be prepared to recognize this challenge very early on and take corrective measures to insure that no team member is left behind. Also, just as in the real world, this may eventually include reassignment of non-performing members. This also becomes part of the Agile content discussion, as team members come and go in the real world as well.

5. CONCLUSIONS

When using the IS 2010 Curriculum Guidelines to refine an IS Application Development program, there may be a desire to limit the number of sequential courses beyond the foundational courses to encourage students to explore more diverse IS content across specializations. This creates a problem of IS content that becomes necessarily repeated (either as original content or as review) as threaded concepts that must be touched on in multiple courses.

Some content, such as Agile methodologies and Application Development, can be discussed thoroughly in any number of discrete courses, but cannot be fully experienced by the students without a true Performance Learning opportunity. To provide this opportunity, two courses such as Systems Analysis and Design and Web/Mobile Programming can be 'paired' in a given semester to create an environment in which a Performance Learning project can be selected. These paired courses could have only individual member students but, ideally, they would have some number of students who co-enroll.

Original course content is focused in both courses within the first quarter so as to lay a foundation for performance for the rest of the semester. Then, Agile concepts are delivered in both courses in a thorough and consistent manner.

Students can then be assigned into Agile teams with representatives from the individual classes and, whenever possible, members who are co-enrolled. In this manner, students can directly experience the Agile concepts at work in fully cross-functional teams.

Care must be taken to be attentive to time conflicts among team members. Additionally, content deliverers who also serve as the Team Leaders must be mindful and actively engaged in the individual team member performance. The content deliverers must ensure that a) co-enrolled students do not experience burn-out nor do they over-step their assigned team roles, and b) students who are struggling do not fall irreparably behind or, if necessary, they may be reassigned as is consistent with real-world Agile implementations.

6. REFERENCES

Cervone, H. Frank (2010). Understanding agile project management methods using scrum. OCLC Systems & Services: International digital library perspectives, 27(1), 18.

Fowler, M., & Highsmith, J. "The agile manifesto." Software Development 9.8 (2001): 28-35.

Mahnic, Viljan (2012). A capstone course on agile software development using scrum. IEEE Transactions on Education, 55(1), 99.

Podeschi, R. (2016). Building I.S. Professionals through a Real-World Client Project in a Database Application Development Course.

ISSN: 2473-3857
v2 n4034

Information Systems Education Journal, 14(6) pp 34-40.

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K., Nunamaker Jr, J. F., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010: Curriculum guidelines for undergraduate degree programs in information systems. Communications of the Association for Information Systems, 26(1), 18.

ISSN: 2473-3857
v2 n4034