Educating Future Software Professionals on Outsourced Software Development

Kevin A. Gary, Gerald C. Gannod, Harry Koehnemann, M.Brian Blake
Division of Computing Studies / Computer Science Department
Arizona State University East / Georgetown University
Mesa, AZ 85212 / Washington, D.C. 20057
{kgary|hek|gannod}@asu.edu / mb7@georgetown.edu

1. Introduction

Software Development is undergoing a not-so-quiet outsourcing revolution. IT tasks, from documentation to customer support to testing, have moved offshore. Development was considered untouchable, in the realm of highly-skilled tasks that require development staffs to be trained and located near business stakeholders. Recent events have clearly shown that cost drivers exert the necessary pressures to invalidate this claim. The challenge then, is for software engineering programs in higher education to construct curricular models that achieve outcomes that include knowledge of, if not some measurable level of competency in, outsourced development best practices. We present a process-oriented undertaking between three campuses, Arizona State University East, Arizona State University Tempe, and Georgetown University, to experiment with learning objectives focused on outsourced development models.

Students in the Division of Computing Studies program at Arizona State University's East campus enroll in a four-semester project sequence called the Software Enterprise that guides them through the full scope of software product development, from business modeling to deployment. Students in the Fall 2004 semester of the Software Enterprise performed Project Inception tasks that produced Software Requirements Specification (SRS) documents and analysis models. These deliverables drive the development of student projects at the three campuses. This provides a vehicle for comparative analysis of development processes based on whether students are co-located with students serving as business stakeholders (ASU East), are not co-located but within a geographic proximity to business stakeholders (ASU Tempe and ASU East), or are geographically separated from business stakeholders (Georgetown University and ASU East). This paper presents existing curricular structures at each campus, describes how these offerings are integrated to mimic outsourcing models, and discusses how these models will be assessed.

2. Software Engineering project offerings at participating institutions

The participants in this distributed, collaborative, outsourced project model are the Division of Computing Studies (DCST) at Arizona State University East (ASU East), the Department of Computer Science at Arizona State University Tempe (ASU Tempe), and the Department of Computer Science at Georgetown University. Each of these programs has a semester or multi-semester project course. In the case of ASU Tempe and Georgetown, existing offerings are

being leveraged to participate in this experiment. DCST at ASU East is in the midst of evolving a single semester offering into a four-semester project sequence. This sequence is the coordination point for the participating institutions. This section describes the project course offerings at the participating institutions, and then describes how these offerings will be leveraged to coordinate an outsourcing model.

2.1 Division of Computing Studies at Arizona State University East

The Division of Computing Studies (DCST) on the Arizona State University East campus is tasked with developing programs in the polytechnic model. Graduating students are expected to be "industry-ready". In the model of a polytechnic, an increased emphasis is placed on hands-on practice over pure scientific study. DCST has responded by offering a new Bachelor of Applied Computer Science program that embodies the polytechnic spirit. A central component of this program is a new four-semester project course sequence dubbed "The Software Enterprise".

The DCST at ASU East created an applied software process course titled "Software Factory" in the Fall of 2001[1]. The initial purpose was to provide a more practical perspective on software development than the comprehensive lifecycle approach taken in many traditional software engineering courses. DCST offers a traditional Software Engineering course and it served as a prerequisite to the Factory course. The factory course, like many software engineering and capstone courses, has students work in teams to solve problems using tools and techniques advocated by two software process used in industry, RUP and XP. The factory course was fairly unique in that all projects had to be developed and deployed for real customers. We reported valuable lessons learned through four iterations of this course[1][2]. We noted that continuity of software projects across semesters was very difficult, yet single semester projects were limited in size and complexity. Students were usually confined to a single role in a project team, if project roles were adhered to at all. It was also difficult to teach process-related material, such as requirements gathering and management techniques, while facilitating a single semester project.

To address these issues, DCST has redesigned the single semester factory course into a four-semester sequence dubbed the Software Enterprise. The curriculum plan calls for two one-year projects that a student participates in serially. This sequence is shown in Table 1.

Semester	*Fall*		*Spring*	
Year	*Course*	*Focus*	*Course*	*Focus*
Year 1	CST315	Tools & Process	CST316	Implement, Test & Deploy
Year 2	CST415	Requirements	CST416	Project / Process Management

Table 1. Student Participation Trajectory in the Software Enterprise

A student entering the Enterprise sequence in Year 1 begins by taking CST315, a Software Tools & Process course. In this course a student gains exposure to a set of tools that support the software process. In the Spring semester of Year 1 students participate actively in the current project instance as developers, testers, and deployers of software by taking CST316. CST415 in Year 2, Fall semester starts a second project release cycle for the students. In this semester the students conceive of new project instances or extensions and develop business cases and software requirements specifications. In the fourth and final semester (Year 2 Spring), students

complete their experience by managing projects to their conclusion in CST416. A key feature of the Enterprise is that students in CST316 and CST416 in the Spring semesters of Year 1 and Year 2 work in project teams together, providing opportunities for mentoring and role playing.

The Software Enterprise structure fits well with an outsourcing model. Software Requirements Specification (SRS) documents created by CST415 students in the Fall may be used as the blueprints for outsourced development teams enrolled in CST316 in the following semester.

2.2 Capstone project at Arizona State University Tempe

The Department of Computer Science and Engineering on the Tempe Campus of Arizona State University offers five courses as part of a Software Engineering concentration track: Introduction to Software Engineering (CSE360), Distributed Computing with Java and CORBA (CSE 445), Software Analysis and Design (CSE460), Software Engineering Project I (CSE461), and Software Engineering Project II (CSE462). CSE 360 provides students with their first group project within this track in the context of a software engineering survey. CSE445 and 450 provide no significant project experience; assignments and projects are standalone learning activities that emphasize specific aspects of their given topics.

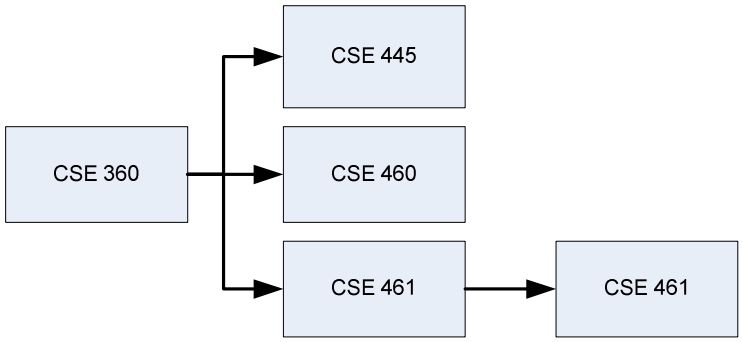

Figure 1. Pre-Requisite Structure for CSE Software Offerings at ASU - Tempe

The CSE461 and 462 courses form a capstone sequence that is intended to provide students with a significant project that is performed over two semesters. The sequence lacks a requirement of continuity in that students may take the courses in non-adjacent semesters. For instance, a student may take CSE461 and 462 in consecutive Spring semesters. In this regard, the sequence has a potential for modeling employee turnover and project re-assignment.

The intention of the CSE461/462 sequence is to have students behave like SEI CMM Level 3 and Level 4 organizations, respectively. Specifically, in CSE461, project teams must achieve a CMM Level 3 standing while in CSE462, project teams must achieve a CMM Level 4 standing. The objectives and outcomes of the CSE461/462 sequence focus primarily upon two topics: providing group experience and providing tool experience. Outsourcing has primarily been limited to separate local groups acting as subcontracts. As such, the real impact of outsourcing has not been experienced in this setting since familiarity with peers that are locally accessible has biased results and in turn made the outsourcing experience merely akin to a large group project.

In the CSE461/462 sequence, students are also required to begin a second iteration (e.g., evolution) of their finished product. Historically, this second iteration has been limited to analysis and redesign although results have varied depending on instructor.

2.3 Software Engineering sequence at Georgetown University

At Georgetown University, Department of Computer Science, software engineering is taught as a three-course software engineering track consisting of Software Engineering I (COSC345) (~18 students), Software Engineering II (COSC346) (~10 students), and Electronic Commerce System Development (COSC545/MGMT630) (~25 students). The three courses are electives for Computer Science majors in both the B.S. track and the B.A. track. COSC345 and COSC346 are junior/senior level courses, while COSC545/MGMT630 is a course jointly offered to undergraduate seniors in computer science and graduate students pursuing an MBA in the McDonough School of Business.

Figure 1 shows an illustration of the major components underlying the courses and how they are related. Software Engineering I (COSC345) concentrates on software design, particularly object-oriented design using multiple methodologies but focusing on the Unified Modeling Language (UML). The course initiates with an introduction to software engineering with in-depth training for object-oriented software design. The second portion of the course provides the students will team-oriented software design experience using realistic problems.

Software Engineering II (COSC346) and Electronic Commerce System Development are continuations to COSC345. Students who take both COSC346 and COSC545 desire a concentrated software experience, however other students decide which continuation of COSC345 is most interesting to them. COSC346 introduces an array of software lifecycle processes in the first few weeks, while COSC545 concentrates on software lifecycles in the distributed computing domain. The remaining twelve weeks of both courses are executed as a real-world development environment, in which students assume specific software engineering roles. The students develop a software product that alleviates a problem solicited from outside companies and organizations. Past problems have come from The MITRE Corporation, Fannie Mae, AEG Capital, and Georgetown University Information Services.

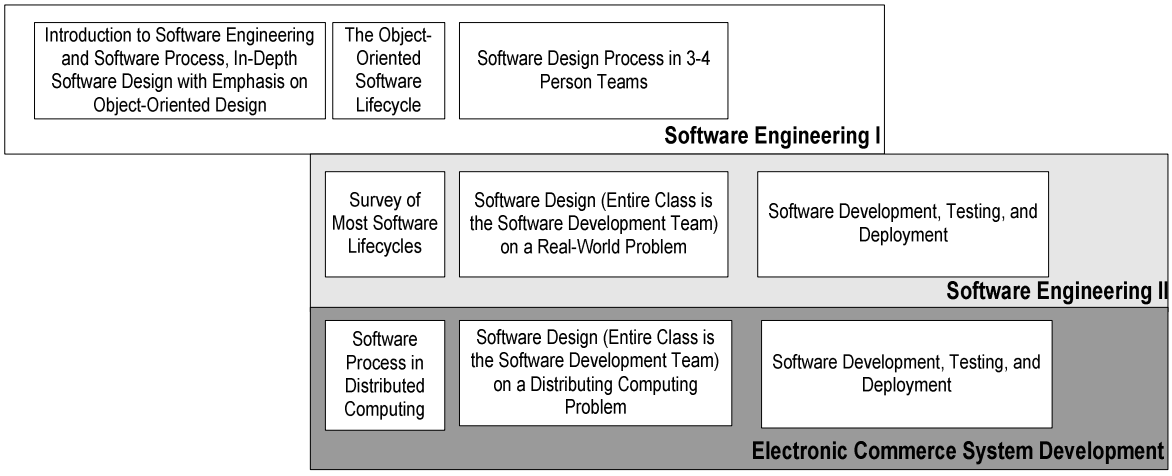

Figure 1. Three Course Software Engineering Track at Georgetown University.

The courses have been extremely useful in instructing students to effectively manage the development software systems in large groups. As shown in Figure 1, students that take all three courses become extremely competent in software design. Of additional benefit are the products delivered to the industry stakeholders. Results from the courses have been reported in earlier work[2][4][5]. The structure of the current software engineering track would seamlessly support providing software designs to other development teams from COSC345 and incorporating software designs from other teams in COSC346 and COSC545.

3. Creating a distributed, collaborative project experience

The outsourcing model we follow uses the CST416 offering (Project/Process Management) of the Software Enterprise at ASU East to coordinate implementation teams from ASU East, ASU Tempe, and Georgetown. Implementation teams from CST316 at ASU East, CSE462 at ASU Tempe, and COSC346 at Georgetown work under the guidance of management teams from ASU East. We will assess and compare the results of the corresponding teams (see section 5).

It should be noted that ASU East and ASU Tempe are geographically separated by approximately 20 miles, and a shuttle service runs between the campuses. Students generally tend to take courses at a single campus, but it is also common for students to occasionally take a course or use the resources of the other campus. Therefore, we are in the unique position to evaluate the consequences of collaborating with a truly remote team (ASU East to Georgetown), a "not-so-remote" team (ASU East to ASU Tempe), and a local team (ASU East to ASU East). Table 2 summarizes these relationships.

Team	*Management*	*Development*	*Characteristic*
Team 1	ASU East	ASU East	Local teams, non-outsourced
Team 2	ASU East	ASU Tempe	Outsourced, but possible to meet in-person
Team 3	ASU East	Georgetown	Outsourced and distributed

Table 2. Summary of Team Relationships between Project Participants

The "remoteness" characteristic coupled with the management role played by ASU East students allows us to mimic an outsourcing model.

Our focus is on assessing the impact of outsourcing on collaborative project experiences, so we are making every effort to keep all other aspects of the project consistent across implementation teams. For example, the implementation teams will use the same SRS document developed by ASU East students from the Fall 2004 semester CST315 course. The implementation teams will also have the same infrastructure support in terms of computing environment and tools. However, it is certainly not possible to control all variables. In particular, students at each of the campuses are at different natural skill levels and have had different academic experiences, so therefore the students themselves become a first-order variable in determining the success or failure of a project.

4. A focus on process

Students must emerge from a "write-a-program-get-a-grade" mentality to a "follow-a-process-produce-a-deliverable" mentality (and eventually to "use-and-improve-processes-to-solve-customer-problems"). This evolution from learner to practitioner is a cultural mindset even at the personal level that should be taught while software engineering students mature in school. We believe the capstone project experience is the best place to provide the mentoring needed to develop this mindset.

As part of this philosophy, the coordination of the outsourced projects centers on software process. The teams are given a set of process practices, tools, and a process meta-model and are then assessed, in part, on their process-related decisions and execution. This aspect is more important than the quality of the deliverable produced, as it more accurately reflects the learning objective (the professional cultural mindset) we want students to develop.

A process *meta-model*[1] is used to constrain process planning and process lifecycle model selection. This keeps major release points in synch across teams, and provides a basis for a higher-level of decision making than exercised by projects constrained to a specific process lifecycle model. Process meta-models considered were the Personal Software Process[6] and Team Software Process[7] (PSP/TSP), Agile Processes (specifically, XP[8]), the Rational Unified Process (RUP[9]), and the Win-Win Spiral Model[10].

Our decision was to use the RUP as the process meta-model, but incorporate aspects of the Win-Win Spiral Model where relevant. Specifically, incorporate risk analysis, risk management, phase boundary planning, prototyping, and negotiation from the spiral model. Though "borrowing" activities from the spiral model, the RUP model is used due to current tool support, currency in the field, definition of a collaborative model with team roles, and inclusion of a Transition phase.

We have decided not to use PSP/TSP and Agile methods for now. PSP/TSP strongly advocates time management and empirical data collection at the personal and small team levels as a basis for personal engineering practices. While this supports our desire to develop a professional cultural mindset in our students, it is simply too burdensome to introduce this process into this environment. The PSP/TSP also seems to lack the flexibility that RUP and the Spiral Model at the process meta-level. Agile methods are also not a fit. There is too much of a reliance on experience and constant integration to provide a suitable framework for student learning of software engineering in an outsourced project setting. Students do not spend enough time on a single course to allow for the daily interactions needed for XP.

One of the major challenges in incorporating process-centered project experiences is determining how much rope to give the student teams. Our experience has been that students must be given some process structure, while at the same time must determine some process definition, enactment, and reporting structures themselves. We want our students to not be just process

[1] The term *meta-model* sometimes takes different forms – "process framework", "methodology", or "paradigm". Process meta-model is used here to describe higher-order process models that may incorporate specific lifecycle models and process practices.

participants, but active process definers. Therefore, we use structures such as RUP to identify process phases (i.e. inception, elaboration, construction, transition) and define responsibilities (or roles) for participants (see RUP workflows), while at the same time allowing students to incorporate best practices from other process models, such as using User Stories for requirements, CRC cards for analysis, or PSP/TSP process scripts for defect identification and tracking. As discussed in the next section, this is a key component of the learning objectives for the project experience.

5. Assessing the outsourcing models

Assessing success or failure of this experiment is multi-faceted and a complex task for the following reasons:

- The learning objectives focus more on soft-skills than on "hard", or technology-specific, skills. It is a difficult task to measure how students have progressed in these areas, though the community clearly recognizes the need to emphasize them more in computing curricula (e.g. Lethbridge[11])
- Though the curricular structures of the three participating programs, as presented in Section2, are similar, they are not the same. Students are participating in the outsourcing experience after having enrolled in non-outsourced versions of prerequisite project courses taught with different syllabi.
- Similarly, student academic and skill levels most certainly vary. For example, students enrolled in CST316 at ASU East are typically sophomores or juniors having completed three programming courses (finishing with Data Structures) and who do not have exposure to distributed and web-based computing techniques. Students at ASU Tempe and Georgetown University do have exposure to these techniques, and may be at the junior and senior academic level.

While such differences cannot be glossed over, these obstacles cannot be removed and we have decided to move forward with awareness of these considerations. We also note that analagous variables exist in the real-world too, as it is certainly the case that outsourced service providers do not come at a "standard" level of competence.

To assess the outcomes of this experiment, we will employ the following techniques:

- An affinity process that provides open-context feedback from students. This process was employed for the latest offering of CST315 at ASU East. This process asks students to respond with short phrases to open-context questions, and then students coalesce results into naturally forming categories and vote on the results. The power of this process is in not pre-structuring feedback according to instructor expectations, but allowing students to converge on a consensus around the value of what they have learned.
- University-mandated student feedback forms.
- Online anonymous student surveys. These will provide the instructors with a means for getting feedback in specific topic areas associated with the projects.
- Student grades. Each program has some amount of historical data to compare against.

- Comparison against non-outsourced versions of the projects. One of the benefits of the curricular structure is that the same project is implemented by teams using the outsourcing models and not using the outsourcing models.
- Anecdotal instructor feedback.
- Project teams conclude process activities by conducting postmortems, and this information may also prove useful in assessing the outsourcing experience.

Finally, we again note that our emphasis is process-centered, not project-centered. While the quality of the resulting software deliverables is a significant factor in the overall success of the project, we emphasize the role of process in determining project success. Were student teams able to define and follow a process? Did students encounter, address, and gain experience with "real-world" project obstacles related to process and people management? The true measure of this experiment's success will be in the answers to questions such as these, and not necessarily in whether their resulting software works at all.

References

[1] H. Koehnemann and K. Gary, "Experiences Using Real Customer Projects for Academic Team Projects", American Society for Engineering Education Annual Conference, 2003.

[2] H. Koehnemann and B. Gannod, "Experiences Using Student Project to Create University Business Applications", American Society for Engineering Education Annual Conference & Exposition", 2004.

[3] M.B. Blake, "A Student-Enacted Simulation Approach to Software Engineering Education" *IEEE Transactions on Education*, Vol. 46, No. 1, pp.124-133, February 2003, IEEE Press

[4] M.B. Blake and T. Cornett, "Teaching an Object-Oriented Software Development Lifecycle in Undergraduate Software Engineering Education", *Proceedings of the IEEE Conference on Software Engineering Education and Training (CSEET2002),* IEEE Computer Society Press, pp 234-241, Northern Kentucky, Ohio, February 2002

[5] M.B. Blake, "Integrating Large-Scale Group Projects and Software Engineering Approaches for Early Computer Science Courses", *IEEE Transactions on Education*, 2005 (to appear)

[6] W.S. Humphrey, Introduction to the Personal Software Process, Addison-Wesley, Boston, 1997.

[7] W.S. Humphrey, Introduction to the Team Software Process, Addison-Wesley, Boston, 2000.

[8] K. Beck, Extreme Programming Explained – Embrace Change, Addison-Wesley, Boston, 2000.

[9] P. Kruchten, The Rational Unified Process – An Introduction. 2nd edition, Addison-Wesley, Boston, 2000.

[10] B. Boehm. A. Egyed, D. Port, A. Shah, J. Kwan, R. Madachy, "A stakeholder win-win approach to software engineering education", Annals of Software Engineering 6, 1998, pp. 295-321.

[11] T.C. Lethbridge, "What Knowledge is Important to a Software Professional", IEEE Computer, pp 44-50, May 2000.

Biographical Sketches

Kevin A. Gary joined the Division of Computing Studies at ASU as an Assistant Professor after spending four years in industry developing software solutions for e-learning. His research interests include automated workflow, distributed software systems, and technology-supported learning. His observations mentoring junior software engineers led him to propose the Software Enterprise at ASU East. Dr. Gary earned his Ph.D. from ASU in 1999.

Gerald Gannod is an Assistant Professor in the Division of Computing Studies at ASU. He received the M.S. (1994) and Ph.D (1998) degrees in Computer Science from Michigan State University. His research interests include software product lines, software reverse engineering, formal methods for software development, software architecture, and software for embedded systems. He is a recipient of a 2002 NSF CAREER Award.

Harry Koehnemann is an Associate Professor in the Division of Computing Studies at ASU where he performs research and teaching in the areas of distributed software systems, software process, and network-enabled embedded devices. Before joining ASU in January of 2001, Harry worked for over ten years as a software developer and consultant. Dr. Koehnemann earned his Ph.D. from ASU in 1994.

M. Brian Blake received the B.S. and M.S. degree in electrical engineering from Georgia Institute of Technology and Mercer University respectively and a Ph.D degree in software engineering from George Mason University. Currently an Assistant Professor of Computer Science at Georgetown University, he has published over 50 technical papers focusing on agent-based systems and software engineering techniques.