A Project Spine for Software Engineering Curricular Design

Kevin Gary, Timothy Lindquist, Srividya Bansal, Arbi Ghazarian
Arizona State University
Mesa, AZ 85212
USA
{kgary, tim, srividya.bansal, arbi.ghazarian} @ asu.edu*

Abstract

Software engineering education is a technologically challenging, rapidly evolving discipline. Like all STEM educators, software engineering educators are bombarded with a constant stream of new tools and techniques (MOOCs! Active learning! Inverted classrooms!) while under national pressure to produce outstanding STEM graduates. Software engineering educators are also pressured on the discipline side; a constant evolution of technology coupled with a still emerging engineering discipline. As a hands-on engineering discipline, where engineers not only design but also construct the technology, guidance on the adoption of project-centric curricula is needed. This paper focuses on vertical integration of project experiences in undergraduate software engineering degree programs or course sequences. The Software Enterprise, now in its 9th year, has grown from an upper-division course sequence to a vertical integration program feature. The Software Enterprise is presented as an implementation of a project spine curricular pattern, and a plan for maturing this model is given.

1. Introduction

Hands-on, or applied learning is expected to produce learners more engaged than those in traditional lecture oriented classes [1]. Significant efforts [2][3][4] in software engineering and computer science education focus on content taxonomies and bodies of knowledge. This is not a bad thing, but taken in isolation may lead educators to believe content coverage is more important than applied learning experiences. There is literature on project-based learning within computing as a means to learn soft skills and complex technical competencies. However, project experiences tend to be disjoint [5]; there may be a freshman project or a capstone project or a semester project assigned by an individual instructor. Yearlong capstone projects are offered at most institutions as a synthesis activity, but to steal a line from Agile methods, *if synthesis is good, why not do it all the time*?

Project experiences, while pervasive in computing programs, are not a central integrating feature. Sheppard et al. [6] suggests that engineering curricular design should move away from a linear, deductive model and move instead toward a networked model: "The ideal learning trajectory is a spiral, with all components revisited at increasing levels of sophistication and interconnection" ([6] p. 191). The general engineering degree program at Arizona State University (ASU) was designed from its inception in 2005 [7] to be a flexible, project-centric curriculum that embodied such integration (even before [6]). Likewise, the Software Enterprise was started at ASU in 2004 as an upper-division course sequence to integrate contextualized project experiences with software engineering fundamental concepts. The computing and engineering programs at ASU's Polytechnic campus merged under the Department of Engineering in 2008, and in 2009 the Arizona

Board of Regents (ABOR) approved a new Bachelor's degree in software engineering (BS SE). The first graduating class will be in Spring 2014 and ASU plans to undergo accreditation review shortly thereafter.

At the course level the Software Enterprise defines a delivery structure integrating established learning techniques around a project-based contextualized learning experience. At the degree program level, the Enterprise weaves project experiences throughout the BS SE degree program, integrating program outcomes at each year of the major. There are several publications on the manner in which the Software Enterprise is conducted within a project course (for example, [8][9]]), and we summarize this in-course integration pedagogy in section 2. The intent of this work-in-progress paper is to describe extending the Enterprise as a spiral curricular design feature we refer to as the *project spine*, and present our plans moving forward to mature and validate this approach.

2. The Software Enterprise Delivery Model

The Software Enterprise is an innovative pedagogical model for accelerating a student's competencies from understanding to comprehension to applied knowledge by co-locating *preparation*, *discussion*, *practice*, *reflection*, and *contextualized learning* activities in time. In this model, learners prepare for a module by doing readings, tutorials, or research before a class meeting time. The class discusses the module's concepts, in a lecture or seminar-style setting. The students then practice with a tool or technique that reinforces the concepts in the next class meeting. At this point students reflect to internalize the concepts and elicit student expectations, or *hypotheses*, for the utility of the concept. Then, students apply the concept in the context of a team-oriented, scalable project, and finally reflect again to (in)validate their earlier hypotheses. These activities take place in a single three-week *sprint*, resulting is a highly iterative methodology for rapidly evolving student competencies (Fig 1).

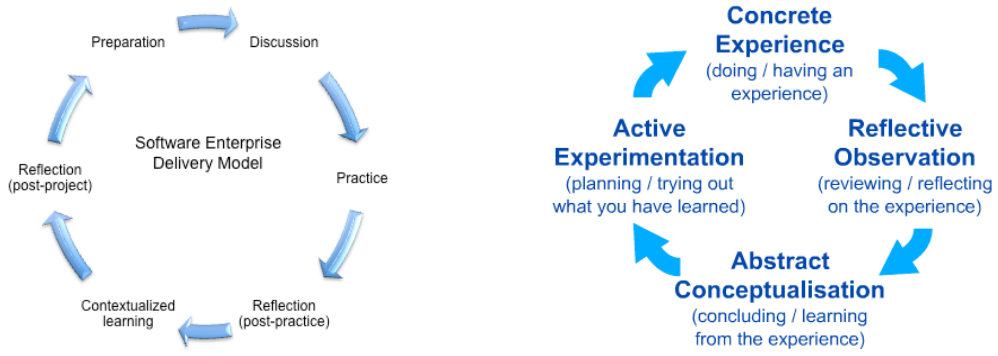

Figure 1. Software Enterprise Delivery (left) Kolb Learning Cycle (right)

The Software Enterprise represents an *innovation* derived from existing scholarship in that it assembles best practices such as preparation, reflection, practice (labs), and project-centered learning in a rapid integration model that accelerates applied learning. Readers may recognize the similarity of the model (Figure 1 left) to the Learning Cycle [10] (Figure 1 right). Our work for the past 9 years in the Enterprise has focused on maturing the delivery process, creating new or packaging existing learning materials to fit the delivery model, and to explore ways to assess project-centered learning.

3. The Software Enterprise Project Spine

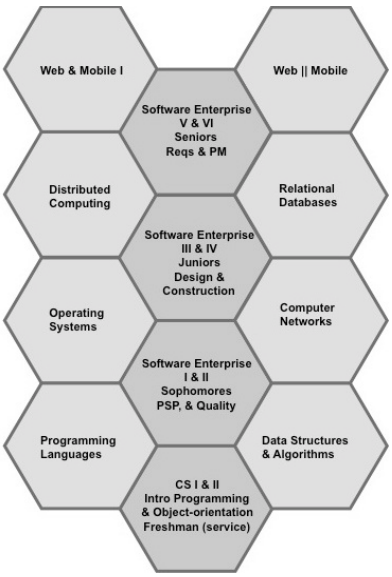

Figure 2. ASU Project Spine

An innovation in the new BS in Software Engineering at ASU has been the vertical adoption of the Software Enterprise. Enterprise courses are now required from the sophomore to senior years. This innovation represents what [6] calls a *professional spine*, as the Enterprise serves as an integrator of learning outcomes for a given year in the major. We refer to our project-centered realization as a *project spine*, where foundational concepts are tied to project work throughout the undergraduate program. There is significant computing literature on projects (embedded, mobile, gaming, etc.) to achieve learning or retention outcomes. However, computing lacks a framework for integrating concepts in a project spine.

The Enterprise is an implementation that moves students from basic comprehension to applied knowledge to critical analysis outcomes. In the BS SE at ASU, program outcomes are described at 4 levels: *describe*, *apply*, *select*, and *internalize*. Students must achieve level 3 (*select* between alternatives) in at least 1 outcome and achieve level 2 (*apply*) in all others. The program outcomes for the BS SE include *Design, Computing Practice, Critical Thinking, Professionalism, Perspective, Problem Solving, Communication, and Technical Competence*. An example leveled outcome description for *Perspective* is given in Table 1. The Enterprise accelerates level 3 outcomes by providing contextualized integrated experiences fostering decision-making in the presence of soft (communication, teamwork, etc.) and hard (technical) outcomes.

Table 1. Perspective learning outcome for the BS SE at ASU

Perspective. An understanding of the role and impact of engineering and computing technology in business, global, economic, environmental, and societal contexts.
Level 1. Understands technological change and development have both positive & negative effects.
Level 2. Identifies and evaluates the assumptions made by others in their description of the role and impact of engineering and computing on the world.
Level 3. Selects from different scenarios for the future and appropriately adapts them to match current technical, social, economic and political concerns.
Level 4. Has formed a constructive model for the future of our society, and makes life and career decisions that are influenced by the model.

The project spine (center of Figure 2 in dark hexagons) integrates technical competencies by assigning projects inclusive of the technical material covered in the regular computing courses. So for example, junior projects (Software Enterprise III and IV) emphasize technical complexities in Networks, Distributed Computing, and Databases, while senior projects emphasize technical complexities in Web and Mobile computing. The technical "focus area" courses are chosen more based on faculty expertise and recruitment goals than software engineering outcomes; one can envision many different areas represented by upper division courses here. These do help address the concern that an accredited software engineering degree has an application area. A risk we have not yet addressed is if the technical area impacts the software engineering process, such as with a soon-to-be-introduced embedded systems focus area.

There are 2 additional aspects of integration to the project spine. As summarized in section 2, the Enterprise integrates software engineering concepts throughout the project experiences. Students in the sophomore year learn the Personal Software Process [11] as a means to build individual understanding of time management, defect management, and estimation skills. They then focus on Quality, including but not limited to testing. In the junior year Enterprise students focus on Design (human-centered and system design principles) followed by best practices in software construction, taken primarily from eXtreme Programming. In the senior year students focus on Requirements Engineering then Process and Project Management. The final aspect of integration is with soft-skill outcomes such as *Communication*, *Teamwork*, and *Professionalism* (see Table 1). Throughout the spine the project experiences are crafted to ensure variations on pedagogy to address these outcomes. For example, in the freshman year students receive explicit instruction in teamwork. In the senior year the emphasis is on formal documentation as a means of communication. In the junior year, students work on service learning projects of high social impact to address communication with sensitive populations in a human-centered context.

4. Literature Review and Next Steps

The newness of software engineering degree programs means there are few studies of program adoption. There are examples of program design and lessons learned [5][12][13], or reflections and recommendations on the software engineering education landscape [14][15][16][17][18]. These are worthwhile guides but do not offer examples on evaluation instruments for program adoption. The SE2004 report [4] contains a chapter on "Program Implementation and Assessment" which discusses a number of key factors in program adoption, but is geared toward accreditation and not evaluation instruments. A survey instrument is presented in [19] but is designed for comparison of a large number of programs and not for adoption challenges. Bagert & Chenoweth [22] survey graduate programs in software engineering but more as an aggregate counting exercise in knowledge areas than as an adoption evaluation approach. The 2009 Graduate Software Engineering project conducted a survey of graduate degree programs [20] and then produced a comparison report [21] of graduate programs to the GSwE2009 reference model, which includes data on program characteristics and in-depth profiles from 3 institutions. A recent study is Conry's [23] survey of accredited software engineering degree programs. Conry summarizes institutional, administrative, and curricular (knowledge area) aspects in describing the 19 accredited programs as of October 2009. Certainly program adoption measures from other engineering programs are also relevant, though software engineering programs are unique due to the forces discussed in section 1.

Our next steps for the Enterprise-as-project-spine involve defining measures for adoption impact, and determining how this concept fits with established patterns for curricular maps in software engineering programs. We plan to use quantitative and qualitative instruments to evaluate adoption. Quantitative data, such as program size, institution type, faculty and student backgrounds, can be collected via available resources (departmental archives or online) and direct surveys. Qualitative data can be collected through survey instruments and interviews of all stakeholders (faculty participants, administrators, and advisors). Different instruments may be used at different times to evaluate "in-stream" attitudes versus post-adoption reflections. Defining and validating these instruments is a significant area of work going forward.

The SEEK, SWEBOK, and CS2013 guides define taxonomies of knowledge in software engineering. Taxonomies are useful and the sign of an emerging discipline. We

have done initial mappings of Enterprise modules to SEEK and CS2013 content areas, and plan to elaborate on these mappings. Specifically, we intend to produce CS2013 course exemplars. Further, the SE2004 report includes a section on program curricular patterns, and we will propose new patterns based on the project spine concept, which we hope will be timely in light of the in-process revisions to the SE2004 [24].

References

[1] National Academy of Engineering. *Educating the Engineer of 2020: Adapting Engineering Education to the New Century*. The National Academies Press, Washington D.C., 2005.

[2] Institute for Electrical and Electronic Engineers Computer Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, 2004.

[3] Association for Computing Machinery & Institute for Electrical and Electronic Engineers Computer Society Joint Task Force (2012). *Computer Science Curricula 2013 (Ironman draft)*. Available at http://ai.stanford.edu/users/sahami/CS2013, last accessed January 9, 2013.

[4] Association for Computing Machinery & Institute for Electrical and Electronic Engineers Computer Society. *Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Joint Task Force on Computing Curricula, 2004.

[5] Shepard, T. "An Efficient Set of Software Degree Programs for One Domain." In Proceedings of the International Conference on Software Engineering (ICSE) 2001.

[6] Sheppard, S.D., Macatangay, K., Colby, A., and Sullivan. W.M. *Educating Engineers: Designing for the Future of the Field*, Jossey-Bass, San Francisco, 2008.

[7] Morrell, D., Roberts, C., Grondin, B. Kuo, C.Y., Hinks, R., Danielson, S., and Henderson, M. "A Flexible Curriculum for a Multi-disciplinary Undergraduate Engineering Degree." Proceedings of the Frontiers in Education Conference 2005.

[8] Gary, K. "The Software Enterprise: Practicing Best Practices in Software Engineering Education", The International Journal of Engineering Education Special Issue on Trends in Software Engineering Education, Volume 24, Number 4, July 2008, pp. 705-716.

[9] Gary, K., "The Software Enterprise: Preparing Industry-ready Software Engineers" *Software Engineering: Effective Teaching and Learning Approaches*, Ellis, H., Demurjian, S., and Naveda, J.F., (eds.), Idea Group Publishing. October 2008.

[10] Kolb *Experiential Learning: Experience as the Source of Learning and Development*, Prentice-Hall, N.J. 1984.

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

[12] Lutz, M. and Naveda, J.F. "The Road Less Traveled: A Baccalaureate Degree in Software Engineering." Proceedings of the ACM Conference Special Interest Group on Computer Science Education (SIGCSE), 1997.

[13] Frezza, S.T., Tang, M., and Brinkman, B.J. Creating an Accreditable Software Engineering Bachelor's Program. IEEE Software November/December 2006.

[14] Hilburn, T.B., Hislop, G., Bagert, D.J., Lutz, M., Mengel, S. and McCracken, M. "Guidance for the development of software engineering education programs." *The Journal of Systems and Software, 49(1999):163-169.* 1999.

[15] Ghezzi, C. and Mandrioli. "The Challenges of Software Engineering Education." In Proceedings of the International Conference on Software Engineering (ICSE) 2006.

[16] Lethbridge, T., Diaz-Herrera, J., LeBlanc, R.J., and Thompson, J.B. "Improving software practice through education: Challenges and future trends." Proceedings of the Future of Software Engineering Conference, 2007.

[17] Shaw, M. (2000). Software Engineering Education: A Roadmap. Proceedings of the 2007 Conference on the Future of Software Engineering, Limerick Ireland, 2000.

[18] Mead, N. (2009). Software Engineering Education: How far We've Come and How far We Have to Go. Proceedings of the 21st Conference on Software Engineering Education and Training (CSEET 2009). 2009.

[19] Modesitt, K., Bagert, D.J., and Werth, L. "Academic Software Engineering: What is it and What Could it be? Results of the First International Survey for SE Programs." Proceedings of the International Conference on Software Engineering (ICSE) 2001.

[20] Pyster, A., Turner, R., Devanandham, H., Lasfer, K., Bernstein, L. "The Current State of Software Engineering Masters Degree Programs", The Conference on Software Engineering Education & Training (CSEET), 2008.

[21] Frailey, D., Ardis, M., Hutchison, N. (eds). *Comparisons of GSwE2009 to Current Master's Programs in Software Engineering, v1.0.* Available at http://gswe2009.org, last accessed January 24, 2013.

[22] Bagert, D.J. & Chenoweth, S.V. "Future Growth of Software Engineering Baccalaureate Programs in the United States", Proceedings of the American Society for Engineering Education Conference. Portland, OR, 2005.

[23] Conry, S. Software Engineering: Where do curricula stand today? Proceedings of the National Conference of the American Society for Engineering Education, Louisville, KY, 2010.

[24] Hislop, G., Ardis, M., Budgen, D., Sebern, M.J., Offut, J., & Visser, W. (2013). "Revision of the SE2004 Curriculum Model." Panel at the ACM Conference of the Special Interest Group on Computer Science Education (SIGCSE), Denver, CO, 2013.