# 2006-1055: SOFTWARE DEVELOPMENT LABORATORY: A RETROSPECTIVE

**Deepti Suri, Milwaukee School of Engineering**

Deepti Suri is an Associate Professor in the Electrical Engineering and Computer Science Department at Milwaukee School of Engineering (MSOE). She primarily teaches courses in the Software Engineering program.

**Mark Sebern, Milwaukee School of Engineering**

Mark Sebern is a Professor in the Electrical Engineering and Computer Science Department at Milwaukee School of Engineering(MSOE) and is the Program Director for MSOE's undergraduate Software Engineering (SE) program.

# Software Development Laboratory: A Retrospective

***Abstract***

*At Milwaukee School of Engineering(MSOE), undergraduate students work on a one academic year (three quarters) Software Development Laboratory (SDL) course sequence in their junior/senior year. SDL was created with a vision of providing a "real-life" team experience to students where they could unite theory and practice while working on large scale ongoing projects in the context of a standardized development process. This paper presents a retrospective on the pedagogical philosophy of the SDL and the specific challenges that we are currently facing in executing this philosophy. This paper is meant to start a debate in the SE education community on whether the issue is the philosophy itself, its implementation or how we are measuring (or not measuring) "success". This paper most likely raises more questions than it answers.*

## 1. Introduction

When the term "Software Engineering" was coined in 1969[10], the software development community recognized the fact that production of software is a complex undertaking. Software Engineering (SE) educators have been struggling since then to provide opportunities in an academic setting where the students can apply SE practice and process to realistic development efforts[13]. According to Humphrey[4] the concept of software process is not so much concerned with particular tools or methodology; instead, the emphasis is on well defined and controlled processes that can be supported by appropriate methods and tools[6]. Software practice, on the other hand indicates the actual artifacts that are delivered while implementing the process.

When student experience with SE process and practice is limited to individual courses, typically by involving internal or external client, they may not gain the desired knowledge and skill. In these courses each team of students typically starts from scratch, constraining the size of the software system that is developed and increasing the risk that the final product may be incomplete or non-functional[13]. There is a general consensus in the SE education community that students need some project experience that extends for more than one academic term[2][12].

An alternative approach implemented by Moore[8][9] in the Real World Lab at the Georgia Institute of Technology, is a project course sequence in which large-scale projects are addressed by a series of student teams over an extended period of time. In this model, less experienced students can work on a large system that is reasonably well defined and documented by previous teams; while more experienced teams can initiate new systems or major enhancements.

Inspired by Moore's vision of creating a "level 5" (referring to the original Capability Maturity Model published by the Software Engineering Institute[11]) software development organization in an academic environment, and encouraged by other educators, the MSOE SE faculty members decided to incorporate a three-quarter (one academic year) software

development laboratory (SDL) course sequence into our undergraduate software engineering curriculum[13]. The initial vision for the SDL was as a software development environment with established processes and procedures, where students could apply skills from earlier classes to produce software in a systematic and repeatable manner, delivering products on time, within budget and of appropriate quality. The undergraduate software engineering program at MSOE[14] whose initial curriculum version included the SDL experience, began operation in 1999 and had its first graduating class in spring 2002. The SE program was visited by the Accreditation Board for Engineering and Technology (ABET) in September 2002 and is one of the first accredited SE programs in the United States. The academic schedule at MSOE is based on a quarter system with three quarters in an academic year. Each quarter involves ten weeks of instruction with the eleventh week devoted to final exams. Typical software engineering courses are three or four credits, and most have an associated laboratory session. The software development laboratory course sequence begins in the winter quarter of the junior year and extends through the fall quarter of the senior year. Upon entry into the software development laboratory, students have already been exposed to the core elements of SE process and practice. These elements are integrated vertically as well as horizontally through out our curriculum. For example, students have already completed courses in programming, data structures, algorithms, design patterns, embedded systems software, requirements, testing and individual software process. During their time in the lab, students simultaneously take courses in software architecture and formal methods.

## 2. How is SDL run currently?

The SDL sequence begins in the winter quarter for junior students. The preparations for it typically start much earlier in the fall quarter. The outgoing seniors typically meet with the junior students two-three times during the fall quarter in order to plan for their transition in to the SDL. Some of the topics that are covered in these presentations are (i) the role of the SDL in the curriculum, (ii) the differences between SDL and the other classes they have taken so far, (iii) the tools/processes currently being used in the lab and, (iv) the current status of various projects and their related technologies.

After these presentations, the juniors complete a survey of their skill sets and preferences. Instructors use this information to form new teams. Team assignments are based on individual preferences, skills and attitudes in an attempt to form diverse and balanced teams. The seniors typically prepare a plan for the new team's first cycle.

The lab processes that are used are based heavily on the Team Software Process (TSP)$^{SM}$ as described by Humphrey[5]. These processes build on student experiences in a sophomore PSP course[4].

We use an incremental delivery model, usually with two development cycles in a quarter. As student teams gain experience, they choose an appropriate length for each cycle. The following table outlines major activities by quarter, for each student cohort.

| Quarter | First Cycle | Second Cycle |
|---|---|---|
| SDL-I (winter quarter, junior year) | • Introduction to lab processes/tools.<br>• Introduction to basic TSP.<br>• Introduction to CMMI – submit a chosen area in which further research will be done[i].<br>• Introduction to post-mortem analysis (Cycle report).<br>• Initiate contact with the stakeholders | • Development of a cycle plan and periodic updates on the execution of the plan.<br>• Introduction to the role of staff group in an organization<br>• CMMI reports<br>• Product deliverables. |
| SDL-II (spring quarter, junior year) | • Getting better with planning.<br>• Focus on processes that work for the project<br>• Incremental deliverables.<br>• Staff activities | |
| SDL-III (fall quarter, senior year) | • Wrapping up major deliverables. | • Transition<br>• Developing a plan for the new team.<br>• Deploying the project. |

As background for consideration of challenges we face in the SDL, it may be useful to describe our current process, tools and student assessment methods.

### 2.1. Process

The SDL typically enrolls 5-6 development teams with 4-6 members in each. Team members are assigned roles adapted from those suggested by Humphrey[5]: team leader, development manager, planning manager, support manager, contact manager, requirements manager and quality/process manager. These role definitions are helpful, especially for new students, in establishing expectations and providing guidance in carrying out the team's work.

To assure continuous improvement of lab processes, we needed a structured way to define, improve and evolve those processes. For this reason, the lab process gives students "staff" assignments in addition to their development responsibilities. For a typical student, who is expected to spend about 10 hours in SDL per week, 30% of that time is earmarked for staff activities.

The instructor meets with each team for a weekly half-hour status update. Prior to each meeting, the team prepares a summary of information like current milestones, actual effort vs. estimate for each team member during the week, earned value of the project so far, milestones to be completed in the next 1-2 weeks, planned times for the coming week and any questions that the team may have. This weekly status update allows the instructor to evaluate team progress on a periodic basis and to intervene immediately if there is an indication of some problem.

### 2.2. Tools

Since SDL students are typically working on large scale projects that span many quarters, they are exposed to various tools for time tracking, version control, issue tracking, defect logging, etc.

For time logging, we use two internally developed tools: Leia (Laboratory Information Engineering Archive) and FAST (Friendly Assessment Support Tools). The Leia tool is based on TSP and currently has limited functionality to manage components and tasks, edit cycle plans, log time and defects and display time logs. FAST is used to track total lab time (task hours + overhead hours) while only task hours are entered in Leia.

We also use CVS, an open source version control system, for configuration management. The lab's Software Configuration Management (SCM) staff group defines and updates "best practices" for configuration management and version control. Mantis, another open source product, is used for defect/issue tracking.

### 2.3. Measuring Student Achievement

Once a team is executing the plan it has made, it is important to assess how things are progressing. This becomes especially important in an academic setting where student assessment is an essential activity. Currently, the student achievement in the SDL is being measured in multiple ways.

Twice during the quarter, students make portfolio submissions to provide evidence that they are achieving the course outcomes[ii]. These submissions are graded on a scale of "Not yet Satisfied", "Minimally Satisfied", "Satisfied" and "More than Satisfied" for each outcome and returned to the students. The students are expected to achieve at least "Minimally Satisfied" for each outcome by the end of the quarter. To ensure that students receive feedback in a timely manner, portfolio submissions are made in alternate weeks by each half of the student groups. All students are provided an option of making a final submission in Week 9, if needed.

To monitor team function, members complete peer evaluations twice each quarter. At the end of each cycle, teams prepare a cycle report, which is a post-mortem analysis of their performance for the cycle. The students are asked to do some critical self evaluation of their team, in the areas of planning, process and quality. They are expected to analyze what worked well for them in the cycle and what did not. Each team must formulate some concrete suggestions on how to avoid recurrence of the mistakes that they made during the cycle.

We have a dedicated web server for the SDL and each team is provided shell access to make their updates. Each team must keep these web pages current to apprise the clients and the instructors of the current state of their project. The team web site contains all product documents (unless protected by a specific non-disclosure agreement), status updates, cycle reports, meeting agendas and minutes, bulletin items, risks and release logs.

## 3. Current Challenges in the SDL

In this section, we present the major challenges that we are currently facing in the SDL, especially in executing in what we believe was the original pedagogical philosophy of the lab.

**Low productivity –** As mentioned earlier, students are expected to spend about 10 hours/week in the SDL. Even when this goal is met, a student spends in a month as much time as a working professional would do in a week. For this reason, the projects in the SDL move at a slow pace and it is very hard to sustain student motivation and client interest.

In the last three years in the SDL, two projects have been brought to a close. One was delivered complete and the other was cancelled. After five years in the lab, the cancelled project was still behind schedule and the client judged that their window of opportunity had long since passed.

Leia (the lab's in-house planning tool) was finally deployed with limited functionality in the lab in winter 2005-2006. This was achieved after some components were developed by instructors and increased project monitoring was provided. The current Leia team continues to deliver additional functionality.

**Lack of Mentoring –** The classes in the SDL sequence are scheduled to meet for four hours each week (typically for two 2-hour sessions). The remaining time that a student spends in the lab is unscheduled. With five development teams in the SDL and only one instructor to go around, a maximum of 48 minutes per group is available. In practice, the instructor usually spends about 30 minutes with each team and uses the rest of the time to handle lab administration issues and conduct lab wide meetings. In the author's experience, this time is insufficient for any kind of active monitoring and mentoring.

**Time Spent by the Students in the Lab-** Analysis of the time logs submitted by students for the last two years shows that an average student spends about 50% of the budgeted time (i.e., 5 hours) in the SDL; for most students the time is spent on the development projects with no significant staff deliverables. Of the actual time spent in the SDL, only half is spent on task-related hours, the rest being overhead. In other words, a typical student spends about 2.5 task related hours in the SDL per week, which is not enough to make any kind of "real" progress on the product.

Deadlines in other classes, unstructured environment of the SDL, no identical assignments/deliverables across the lab, and time wasted in context switching are some of the reasons that are offered by the students for their performance in the SDL. Many students have suggested we schedule longer lab sessions and ensure that the instructor is available. With such a scheme, the team would not have to work around individual scheduling issues. They also believe that less time will be spent on context switching. While the suggestion has merit, academic budget constraints limit instructor time for a 3-credit course and overall class scheduling requirements restrict flexibility.

**Inadequate tools -** Various tools that are used in the lab are not adequate for the tasks that our development processes require. For example, presently we have no automated way to calculate a project's earned value, productivity, yield, and other process quality indicators (PQI's)[5]. These parameters currently have to be calculated manually and the process is cumbersome at best. The lack of standardized data makes it difficult for a team to see how they are performing relative to the other teams in the SDL. If this data were easily available, it would facilitate the teams in sharing their "best practices" with each other.

**Choice of projects** – We have always tried to get SDL projects from external stakeholders. Because of the slow development pace, these projects also tend not to be "on the critical path" for the sponsor. Since the stakeholders also tend to be very busy, they may not interact with the students and display their enthusiasm for the project on a continual basis. The students tend to interpret this lack of participation as meaning that nobody is waiting to use the product that they are working on. This tends to be a huge demotivator.

**Unfamiliar Technologies –** In some instances, the projects that are being developed in the lab are using products and technologies that the instructors as well as the students are not fluent with. In that case, it becomes harder for the students to get help when they need it.

**Grading –** As in other courses, the grading criteria for the SDL sequence are made available to the students on the first day of class. Unlike other classes, students have a much harder time calculating their grades based on the feedback that they receive from the SDL instructors and hence are frustrated. Whether we like it or not, grades continue to remain one of the biggest motivators for all students in the classroom. The grading criterion for SDL-I for winter 2005-2005 has been included in the table below.

| | |
|---|---|
| Team process performance and improvement | 20% |
| Development team work products | 40% |
| Individual portfolio submissions | 25% |
| Lab Process Improvement Report | 15% |
| Total | 100% |

## 4. Process Improvements

Some incremental process improvements have been made in the SDL based on the feedback that students have provided. Some of the changes that have been made are:

- Additional instructor help – The winter quarter of 2005-2006 was the first time that the SDL had 6 development teams. There are now two instructors in the lab and each instructor has prime responsibility for 3 teams. All portfolio submissions are graded by both instructors and combined feedback is provided to the students.

- Increased and improved communication with the stakeholders – We have requested that our stakeholders interact with their teams on a weekly basis, with at least one face-to-face meeting every two weeks. This has already started to show positive effects.

- More structured weekly deliverables – Efforts are being made to make sure that each team member has structured weekly deliverables. We also seek to balance student workloads with other major courses in the same quarter.

- Simplifying web page updates – Processes have been put in place so that the documents that are checked into CVS are automatically exported to the web. This ensures that updates to documents from the project's web site become available without manual intervention.

- Improved and detailed guidelines on the grading criterion – Special efforts are being made to address grading criterion and concerns during the weekly lab wide meetings.

## 5. Summary

The Milwaukee School of Engineering has one of the first ABET accredited SE programs in the United States. We believe that the strength of our SE program lies in its innovative curriculum that, while incorporating all the core elements of software engineering practice, also places special emphasis on software engineering process. Presently, there exist numerous challenges in executing our philosophy for our Software Development Laboratory (SDL). The problem could be (i) in our vision of the SDL where the initial goals that we set for ourselves may be unrealistic (ii) in the resources that have been allocated for the lab (iii) our choice of projects (iv) our expectation of students' or any combination of these. We have presented the current state of affairs in an effort to share our experiences, successes and challenges with other SE educators. We hope that they will do the same and many collaborative opportunities will arise as a result.

## 6. References


[1] CMMI Product Development Team, "*CMMI[SM] for Systems Engineering/Software Engineering/Integrated Product and Process Development, Version .02: Continuous Representation*", Technical Report CMU/SEI-2000-TR-031, Software Engineering Institute, Pittsburgh, PA, USA, 2000.

[2] Groth, P. G. and Robertson, E.L., "It's All About Process: Project-Oriented Teaching of Software Engineering," *Proceedings of the 14th Conference on Software Engineering Education & Training*, Charlotte, NC, USA, February 2001, pp. 7-17.

[3] Halling, M.A., Zuser, W. et. al. , "Teaching the Unified Process to Undergraduate Students", *Proceedings of the 15th Conference on Software Engineering Education and Training*, Covington, KY, USA, February 25-27, 2002, pp. 148-159.

[4] Humphrey, W. S., *A Discipline for Software Engineering*, Addison-Wesley, 1995.

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

[6] Hunter, R.B. and Thayer, R.H. (eds), *Software Process Improvement*, IEEE Computer Society, 2001.

[7] Lisack, S., "The Personal Software Process in the Classroom: Student Reactions (An Experience Report)", *Proceedings of the 13th Conference on Software Engineering Education and Training*, Austin, TX, USA, March 6-8, 2000, pp.169-175.

[8] Moore, M. M. and Brennan, T., "Process Improvement in the Classroom," *Proceedings of the 8th SEI CSEE Conference, Lecture Notes in Computer Science*, New Orleans, LA, USA, Springer, March/April 1995, pp. 123-130.

[9] Moore, M. M. and Potts, C., "Learning by Doing: Goals and Experiences of Two Software Engineering Project Courses," *Proceedings of the 7th SEI CSEE Conference, Lecture Notes in Computer Science*, Springer-Verlag, January 1994.

[10] Naur, P. and Randell B. (eds), *Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee*, NATO, 1969.

[11] Paulk, M. C., Curtis, B., Chrissis, M. B., and Weber, C. V., *Capability Maturity Model[SM] for Software, Version 1.1*, Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, Pittsburgh, PA, USA, 1993.

[12] Port, D. and Boehm, B. "Using a Model Framework In Developing and Delivering a Family of Software Engineering Project Courses," *Proceedings of the 14th Conference on Software Engineering Education & Training*, Charlotte, NC, USA, February 2001, pp. 44-55.

[13] Sebern, M.J., "The Software Development Laboratory: Incorporating Industrial Practice in an Academic Environment", *Proceedings of the 15th Conference on Software Engineering Education and Training*, Covington, KY, USA, February 25-27, 2002, pp. 118 - 127.

[14] Sebern, M. J. and Lutz, M. J., "Developing Undergraduate Software Engineering Programs," *Proceedings of the 13th Conference on Software Engineering Education & Training*, Austin, TX, USA, March 2000, pp. 305-306.

[15] Suri, D. and Sebern, M.J. , "Incorporating Software Process in an Undergraduate Software Engineering Curriculum: Challenges and Rewards", *Proceedings of the 17th Conference on Software Education & Training*, Norfolk, Virginia, pp 18 - 23, March 2004.

---

[SM] Personal Software Process, PSP, Team Software Process and TSP are service marks of Carnegie Mellon University.

[i] During the first quarter of the lab, each student is required to write a paper on one Key Process Area (KPA) of the Software Engineering Institute's Capability Maturity Model [1] and to suggest a specific way that the chosen KPA could be implemented in the lab. A number of these proposals become the basis for various future staff team assignments.

[ii] Currently the outcomes for all the courses in the SDL sequence are the same and are available at http://www.msoe.edu/eecs/cese/courses/courses.php3?course=SE-3091