

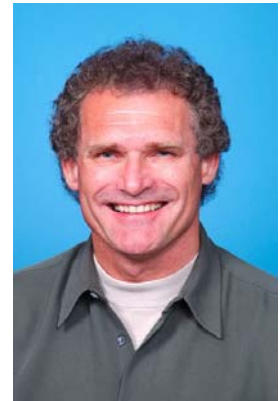


February 2005 Issue

Why Invest in Software Engineering Education?

Authored by Kal Toth, Portland State University

Edited by Ed Carroll, ProDX



Abstract: This is the first article in a series on professional development programs, including the benefits and value proposition for both students and their companies. These articles will draw on the Oregon Master of Software Engineering (OMSE) program at Portland State University (PSU).

Introduction

This first article sets the context for subsequent articles expanding upon particular sub-topics including scope management and outsourcing, criteria for selecting suitable software processes, guidance for applying software risk management, and an assessment of the relative merit of a range of software estimating techniques. These articles will draw on materials, discussions and case studies explored within the Oregon Master of Software Engineering (OMSE) program (<http://www.omse.pdx.edu>) at Portland State University and affiliated with the Oregon Graduate Institute (OGI).

Recent SAO articles have explored a range of emerging software processes and methodologies, software process improvement strategies, and approaches for enhancing software quality. These articles were meant to stimulate interest, provide insights, and motivate readers to consider actions that could enhance the productivity and quality of their software products and services.

But what is a company to do if current practices and personnel are not yet up to the task? Or what if a few employees are convinced of the benefits but do not have sufficient depth, arguments or credibility to sell the concepts and thereby achieve buy-in with skeptical developers or technical managers?

Clearly, some perspiration must accompany inspiration to achieve tangible and lasting results. Investments in education and training can be among the investments a company can take to turn the corner.

But fiscally constrained companies need not only to get continuously better, but they need to do so while minimizing the cost of professional development. And this is not

only with respect to direct education costs, but also in terms of time lost to ongoing projects while personnel are busy studying. Therefore such professional improvement programs need to minimize organizational impacts while offering concrete results, aligning with business priorities, and delivering just-in-time knowledge and know-how.

Furthermore, such education and training should not just inform best software engineering practices, but should also offer convincing return on investment rationale and arguments to senior management to gain corporate commitment for the necessary budgets and release times.

Last but not least, such programs must provide practitioners with various forms of “payoff”. After all, they will be devoting considerable sweat equity and time away from family to learn and facilitate changes while doing their best to meet their usual company obligations. Given current fiscal and job-mobility constraints, software professionals are more focused on expanding opportunities within their firms than looking elsewhere. By devoting energy to enhance their own capabilities, they will find and develop more interesting and challenging projects within their companies and simultaneously build respect and credibility with their peers and managers.

Can SE Education and Training Increase Software Productivity and Quality?

After years of systems and software technology analysis, modeling, design and implementation work, I was asked by medium and large sized companies to move into senior technology positions to help solve technical and management problems and fill gaps. The common complaints had to do with poor software productivity and software quality. It was the norm for project schedules and budgets to overrun and far too much time was being spent fixing brittle software long after delivery. Poor estimating and lack of scope control were determined to be the root cause.

I soon launched software process improvement initiatives across these companies through awareness seminars, process frameworks and training sessions. Customer satisfaction and a total quality focus were reinforced by in-depth training in critical process areas. This motivated the troops and provided them with new knowledge, tools and ways of practicing their craft. Key people were groomed to become change agents yielding pockets of improvement - a few stellar projects resulted. Naturally, these were promoted with customers and new recruits alike and some additional converts were acquired.

Many managers and technical people were broadly introduced to the initiative – they politely listened and agreed to the ideas but were not given enough insight or direction to take any tangible action. Only a relatively small number were explicitly trained to learn how to improve their practices. Those who were neither introduced to the processes nor trained in them tended to resist change altogether. Therefore many of the projects continued along as before, with no noticeable changes or improvements. So the results were mixed. However, the successes were encouraging and provided good evidence that effective education and training will lead to constructive changes and improvements.

These experiences reinforce that change takes time and change needs commitment from everyone involved – senior management as well as software developers across the board. I conclude that investments in software engineering need to be made across the company to achieve significant, ongoing and long-lasting improvement.

However, such investments need not be made using a “big-bang” approach. A preferred approach would be to develop a phased and incremental action plan that starts with critical departments and smaller projects, and then progressively covers other departments and projects. Whatever the initial coverage of such a deployment plan, directly related senior management and software personnel should support and be involved as follows:

Senior and Middle Management: Should not only allocate budgets to education and awareness, but also ask the right questions and facilitate their SE personnel to ensure best practices are indeed enabled and practiced. In most cases I believe this will also require increased software engineering awareness on the part of senior and middle management.

Key SE Personnel: The Company needs to ensure that suitably motivated and experienced software engineers complete selected SE courses or a complete program, depending on their roles and responsibilities. As change agents, they should be respected software developers or managers. Key SE personnel should be expected to devote a portion of their time to mentoring and coaching more junior personnel.

Other SE Personnel: To facilitate change and to gain buy-in, it is essential, that all associated personnel be made aware of the processes, benefits and investments they will need to make to help the company move forward and improve software engineering as practiced across the company. In-house professional development seminars and tailored short courses can achieve this objective cost-effectively.

What Should SE Education and Training Programs address?

Clearly, traditional computer science, computer engineering and information technology programs have not been preparing students sufficiently to become truly effective practitioners in the software industry. Learning on the job has been taking too much time and does not guarantee project success. In fact, the current lack of mobility in the industry has the potential of creating retrogressive software technology cultures of increasing mediocrity within companies.

Recognizing this, about five years ago I decided to actively attack perhaps the most important root cause of software project failures – certain critical deficiencies and gaps in the competencies of software developers and software managers. I began to support programs that ranged from tailored professional development courses on customer sites, through continuing education diplomas and certificates, to advanced masters-level graduate programs in software engineering.

I soon confirmed that students joining these programs are thirsting for learning in how to develop software “right or better”. They want to increase their value, naturally enough. But they also want to help their company’s get better and open opportunities for more challenging projects. They firmly believe that they can add value, reduce overruns, improve productivity and quality, make better estimates, set achievable goals, and avoid big disappointments. Perhaps most of all, they want to increase respect and credibility within their companies and among their peers, and get recognized and listened to as professionals.

It is evident to many of these students that cultural change is necessary within their companies for significant or breakthrough results. The more senior they are, the better their chances of influencing their managers to support needed changes.

Companies that have managed to put several of their students through these programs report that they have managed to promulgate software engineering concepts and thinking throughout their corporate cultures and have achieved significant improvements.

Here is a sampling of improvements most often reported:

- Improved management of scope through increased focus on specifying and controlling requirements and consistently reviewing them with customers, users and marketing.
- Better project planning and more attention to tracking schedules, effort and budget.
- Earlier detection of problems through regular peer reviews, design walkthroughs and software inspections of critical code.
- Increased maintainability through consistent adherence to coding standards and the production of design documents using standardized notations and descriptions.

I have also interviewed several senior technology vice presidents and managers over the years. They have consistently identified the following software engineering competencies needing critical attention:

- Better software estimating to increase confidence in schedules and budgets.
- More comprehensive and automated testing to reduce defective software in the field.
- More thorough identification and tracking of risks to avoid nasty surprises.
- More effective communications and negotiation skills to deal better with difficult team, customer and contractor relations.

In addition to these, I see other competencies that software engineers should be developing to address emerging technology challenges:

- Component-based development frameworks for reuse and product integration.
- Outsourcing competencies to better manage off-shore subcontracts.
- Formal SE methods for specifying and checking correctness of critical software.
- Decision making techniques to support critical decisions such as “Make vs. Buy”.
- Product-line techniques for streamlining the evolution of software product lines.

Can SE Education Be Accomplished without Negative Business Impacts?

My interviews with software companies have reinforced company concerns about time away from projects. Senior managers in virtually all companies recognize that investment in education and training is critically necessary to ensure that the company stays ahead of the competition and that they retain key personnel. However their most important software specialists and managers are already heavily loaded and are continuously tied-up maintaining project momentum to deliver products on time.

Hence companies want SE programs that address the following:

- **Minimize interference** (between company and education schedules): Evening and weekend classes are preferred. Many, but not all, companies are very willing to allocate a part of office time to education and training recognizing that their trainees are often putting in 2 to 4 times as many hours to complete readings and assignments.
- **Easy Access:** Both companies and students are interested in convenient access to classes and materials to minimize lost travel time. Thirty minute drive times are desirable. Web enabled access and email are also important. Several companies are interested in on-line web-enabled course delivery including support for on-line discussions and streamed lecture materials.
- **Leveraging Company Projects:** Several companies support the idea of letting students apply concepts learned in class to company projects (past and present presumably). Intellectual property rights could be a concern, of course, but this can be managed in most cases.
- **Immediate Applicability:** Flexibility in the order in which courses can be taken is highly desirable. This allows students to register in courses that deliver knowledge and competencies just-in-time and thereby yielding immediate benefit while reinforcing the concepts learned in class.

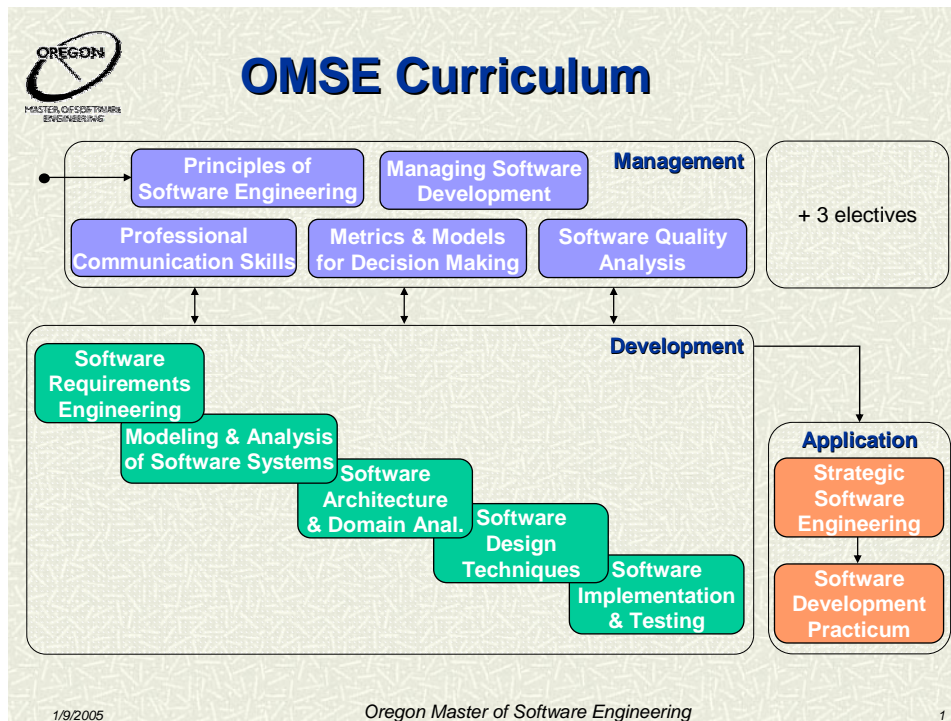
Oregon Master of Software Engineering

Several programs across the United States support many of the program requirements expressed above including those at [Carnegie Mellon University](#), [Seattle University](#) and [George Mason University](#).

In Oregon, the [Oregon Master Software Engineering](#) (OMSE) program was established in the late 90's to address the above requirements. OMSE courses may be taken at Portland State's main campus downtown and the Capital Center in Beaverton. The OMSE program balances software management, software development and the application of SE processes. Students may take individual courses for just-in-time professional development, 3-5 courses comprising a software engineering certificate, or all the SE courses plus approved electives for the complete master of software engineering.

The program builds on foundation knowledge and industry experience of students who must have a minimum of two years of direct software-related experience. The program leverages the knowledge and experience of its faculty, all of whom have in-depth software industry experience. OMSE uses an active learning approach that encourages

discussion in class about emerging problems as well as the problems the students are



addressing on the job. Students thereby learn from each other as well as from the faculty and the course materials.

The courses make use of web tools to facilitate access to materials and active collaboration among students and with faculty. Courses are offered once a week 3 hours per evening or over Friday afternoons

(5 hours) and Saturdays (5 hours) over three “weekends”. For more information about OMSE see <http://www.omse.pdx.edu>.

About the author

Dr. Kalman (Kal) Toth, Associate Professor in Portland State University's Department of Computer Science, is the Associate Director of the Oregon Master of Software Engineering (OMSE) program. He delivers courses on software engineering principles, software project management, software design techniques and the program's software engineering practicum. He has over 25 years of technical, consulting and management experience in large and small technology companies including Hughes Aircraft, Datalink Systems Corp., BC Software Productivity Centre, and the CGI Group Inc.

He may be contacted at ktoth@cecs.pdx.edu or see <http://www.cs.pdx.edu/~ktoth/>.

For more articles visit www.sao.org/newsletter