An Academic Perspective on Globalization in the Software Industry

Umesh Bellur
umesh.bellur@iitb.ac.in
School of Information Technology, IIT Bombay
Powai, Mumbai 400076, India

I. INTRODUCTION

While "outsourcing" has come to be accepted and even expected when talking of software development and business process automation, little thought appears to have been given to packaging the experiences gained so far in practicing globalized development to extend the software engineering curriculum at academic institutions. Having garnered over a decade of experience in all sizes and shapes of such projects, it is time we institutionalized these practices at the academic levels. In this article I will briefly explore the issues brought to bear on the software engineering process[1] by globalization and then delve into a possible curricular change for students of computer science and information technology studying software engineering.

II. ISSUES IN GLOBAL SOFTWARE DEVELOPMENT

There are many cooperation models we can follow in doing distributed globalized software development depending on the degree of separation [2]. These range from separate teams in independent companies to a single team of one company distributed over multiple locations. Each of them promise different benefits and imply different challenges. In this article we are not focused on any one of them but are going to look at an abstraction of distributed development that can subsume any of these models. As is to be expected there are several classes of issues that must be considered when talking about global software development.

- **Strategic** Picking the right strategy for global development is critical. We have already covered some of the models that can be adopted and one of these or a variant needs to be decided on. The secondary issues involve putting the appropriate management teams in place and this is influenced by sociocultural aspects as well. Lastly comes the strategy for chunking the work and distributing it. This could be driven on the basis of functionality, localization or development stages[3].

- **Technical** By technical issues I mean the way a system is architect ed and decomposed into it’s constituent parts. Just as a monolithic piece of software is not amenable to distribution during development, so is a componentized system with a high degree of coupling. The situation can get worse with circular dependencies and particular attention needs to be paid to this. The introduction of service orientation has helped tremendously but system decomposition is key. Another angle to deal with here is that of tools needed for development that can help with data availability, collaborative editing and change control.

- **Sociocultural** Non technical issues dealing with cultural and social differences between geographic locations considerably complicate a distributed development effort. A pleasant working relationship between members of a team is a must if progress is to be guaranteed and we cannot underestimate the effects of understanding and adapting to different cultural makeups.

- **Process and Management** Managing a global software development process has many challenges such as knowledge transfer, quality assurance and configuration management. Knowledge here will take many forms including application knowledge, development standards, technology knowledge and company culture knowledge. KM Technology can considerably ease the pain of sharing information but must be deployed early and broadly.

- **Communication** The factors of language and time figure into the communications challenge. Especially in projects that may need a language specific user interface. Differing time zones may at first glance appear inconsequential but temporal dispersion does add to logistical challenges such as arranging or a video conference across teams.

A review of these issues clearly points out the need to specifically consider the effects of globalization on process and product aspects of software engineering[4]. However, to date, we have not witnessed the abstraction and institutionalization of the ideas borne out of experience in academic curricula even though scattered efforts towards this have take place[5]. The next section outlines an attempt to do so.

III. PERSPECTIVES ON ACADEMIC IMPACT

The standard undergraduate or introductory graduate course on software engineering offered by most computer science and engineering departments in the world today is centered on two aspects:

- **Process and**
- **Product**

Both of these aspects target traditional software development practices which is primarily single site development. Another unfortunate aspect is the theoretical nature of how these
courses are conducted - rarely are real projects offered and accomplished in a professional manner. This makes these courses an easy target for most students who wish to earn credits without much effort. The result is a graduate who is hardly able to estimate effort, make a realistic project plan or drive development of a project of any complexity. Addressing this deficit in academia cannot be done one step at a time - rather we have to attack content AND methodology together in a coherent offering that will produce “real” software engineers who can deal with development in today’s context.

IV. Toward a Relevant Software Engineering Curriculum

In the spirit of lending academic coursework a practical twist, I will focus on two changes that must be made:

1) Change the traditional course into a lab in conjunction with the local industry. The focus here must be to expose students to real software projects with real stakeholders.

2) Bring in the global perspective via collaborative software engineering education involving multiple universities dispersed geographically across multiple time zones.

Needless to say, we need to incorporate our findings regarding globalization of software development into the theoretical aspects of process and product as well.

At IIT Bombay, we are offering a semester long software engineering course for graduate students that aims to address these issues. In this course, teams of 4-5 students are mentored through an end-to-end experience with an industry-sponsored project and are expected to adapt and execute a software engineering methodology in their project. The major objective of this course is to ready students for the marketplace by requiring them to exercise the knowledge they have gained in previous software and software engineering courses on industry-specified real world problems. Specifically, upon completing this course, students are expected to have applied their software engineering knowledge to aspects of:

- Software design driven by non-functional requirements such as scalability, security, usability and performance, generating and evaluating design alternatives and pattern-based software design
- The process of developing software, such as configuration management, project management, planning, team structure, roles and responsibilities, incremental and iterative, workbook-centered, object-oriented, agile, lean software development, project planning
- Presenting work to an audience of peers, and
- Business issues in software development

The external organization typically assigns one or more of their personnel to work with the students - mostly in the provision of requirements, but also as mentors. Finally, industry representatives are invited to the mid-term and final presentations made by students, to act as the "instructor-for-a-day", and provide feedback on how the students are proceeding, with specific feedback on what portions of the projects could make a suitable employment portfolio.

The other interesting aspect of this course will be the focus on global software development. To this end, the course will run in parallel with another similar course at Ohio State University, Columbus. We propose to extend these courses to include projects conducted jointly by students at IITB and at OSU, and involving industries both in the Columbus and Central Ohio and the Mumbai regions. Every project undertaken by students will have a global development team spread over these two universities and work together as a single entity to accomplish the project. The industry interaction will primarily be with the local team even though we have identified a source of funds for the teams to interact with each other remotely and in person.

V. Conclusions

In this paper, I have presented the challenges in global software development and then identified the gaps in the academic curriculum that should be addressed in order to train software engineers to be cognizant of current issues in executing large, global software projects. We have also outlined a practically oriented academic course that will stress on the skills needed to actually execute global projects. Over multiple offerings of such a course we hope to abstract out the actual issues faced and present suitable solutions as learnings.

REFERENCES