World Transactions on Engineering and Technology Education *© 2018 WIETE*
Vol.16, No.3, 2018

Quality development (QDev) unit in a software engineering school

Hadas Chassidim, Dani Almog & Shlomo Mark

Shamoon College of Engineering
Ashdod, Beer Sheva, Israel

ABSTRACT: Addressing software quality and testing as part of the software engineering education programme has become more important as a result of new software development trends, such as incremental development methodologies (e.g. Agile), continuous integration (CI) and continuous development (CD). In this article, the authors present how to integrate these important topics into software education. The main debate has been whether software quality is a specialisation within the software engineering profession or must be an integral part of the engineer's training. In the authors' view the debate already has been settled by industry, through the new trends and evolution of the development lifecycle. It is necessary that software engineers be familiar with, and address, software quality and testing. The aim of authors of this article was to examine the current teaching curriculum, and suggest the tools and means for facilitating integration of quality and testing into the software engineering school programme.

INTRODUCTION

Software is eating the world are the words of Marc Lowell Andreessen, American entrepreneur, investor and software engineer, co-founder of Netscape and co-author of the Web browser, Mosaic. Andreessen described the expanding direction of software engineering (SE), the transition from monolithic systems to micro-services and unexpected events caused by the complexities of environments. Today, the infrastructure can be viewed as code that connects hardware, software and services by cloud computing. This innovative trend shortens development time and costs, but increases complexity, and introduces new challenges to quality and testing by software engineers.

There is no doubt that these challenges call for research and development to enable the implementation of the dramatic changes, with a special emphasis on software quality and testing. These needs also should be addressed in software engineering schools that qualify the future engineers. Academia is committed to developing the students' capabilities, including their technical and soft skills to successfully solve the problems encountered. However, the current curriculum does not necessarily adopt this approach. The authors believe that the variety of processes required should be integrated with the mandatory and elective courses, such that the quality and testing layers should be consolidated in the curriculum from day one rather than inserted artificially later, where they are separated from the development process. Hence, the proposal is to facilitate a designated unit that is responsible for the implementation and the development of all the quality and testing needs.

Software Engineering Trends

Industry adapts to rapid development approaches, including continuous delivery and adjustment of software artifacts, and responding to rapid requirements changes; the transition to microservers, telemetry data and users' experience [1][2]. Therefore, there is a large investment in technologies and tools that support software construction. Particular emphasis should be placed on approaches that improve quality or expand the reach of SE into new domains. The emergence of new technologies allows quality improvements based on empirical studies of processes and tools that deliver concrete, actionable results, with measurable benefits supported by methods such as knowledge management (KM) [3].

BACKGROUND

Quality and Testing Drivers

Quality in SE is a wide discipline that deals with all the stages of the development process and related activities. It starts with requirements engineering, thorough analysis and design, verification and validation. It continues with deployment, aftermarket interoperability, maintenance and retrospective review [4]. Software engineering has long been studied in both academia and industry. However, the fundamental changes in SE require adjusting traditional methods and tools to meet the new needs. The development of solutions in the field often precedes the research or alternatively solutions are lacking due to rapid development and constraints of the demanding market. Many times, quality aspects are neglected due to wrong perceptions, low prestige and lack of knowledge.

Latest development directions require the integration of quality and testing activities throughout the development process, and not only after deployment. The approach presented in this article focuses on the quality and testing implications in advanced development environments, viz. continuous integration (CI), continuous deployment (CD), and development and operations (DevOps) [5]. Also considered are non-functional aspects, such as privacy and the user's interaction with the system to ensure its usability.

These new software methodology trends involve the development of dedicated skills and techniques that support continuous practice. From an academic point of view, there is an opportunity to examine the curriculum, to check whether future software engineers are sufficiently equipped with the required knowledge, abilities and skills. Revised curriculum guidelines help university faculties create or update undergraduate software engineering programmes [6].

Software Engineering Curriculum

The standard SE curriculum is usually inspired by two sources of domain knowledge: software engineering body of knowledge (SWEBOK) [7] and software engineering 2014 - curriculum guidelines (SE2014) [6][8]. The SWEBOK and SE2014 propose that SE programmes should include:

- Professional knowledge: show mastery of software engineering knowledge and skills, and of the professional standards necessary to begin practice as a software engineer.
- Technical knowledge: demonstrate an understanding of, and apply, appropriate theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification and documentation.
- Teamwork: work both individually and as part of a team to develop and deliver quality software artifacts.
- End-user awareness: demonstrate an understanding and appreciation of the importance of negotiation, effective work habits, leadership and good communication with stakeholders in a typical software development environment.
- Design solutions in context: design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal and economic concerns.
- Perform trade-offs: reconcile conflicting project objectives; find acceptable compromises within the limitations of cost, time, knowledge, existing systems and organisations.
- Continuing professional development: learn new models, techniques and technologies as they emerge, and appreciate the necessity of such continuing professional development.

The software engineering education knowledge (SEEK) survey shows that software processes (PRO), software quality (QUA) and software security (SEC) are the major factors that affect most areas of SE, as shown in Figure 1. The recommended teaching time for each topic appears in this survey, whereas software quality is treated as a pervasive background topic, rather than a separate topic.

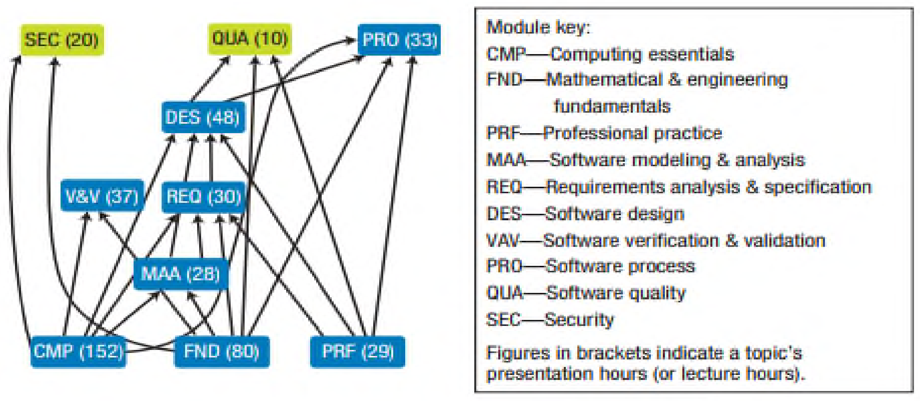
SEC (20)
QUA (10)
PRO (33)
DES (48)
V&V (37)
REQ (30)
MAA (28)
CMP (152)
FND (80)
PRF (29)
Module key:
CMP—Computing essentials
FND—Mathematical & engineering fundamentals
PRF—Professional practice
MAA—Software modeling & analysis
REQ—Requirements analysis & specification
DES—Software design
VAV—Software verification & validation
PRO—Software process
QUA—Software quality
SEC—Security
Figures in brackets indicate a topic's presentation hours (or lecture hours).

Figure 1: Current knowledge unit sizes, interdependencies and suggested teaching times for software engineering education knowledge (SEEK).

A further analysis regarding the knowledge units (KUs), including breaking down each KU into knowledge areas is described by Ardis et al in SE2014 [6]. This leads to two observations:

1. Although software quality (QUA) is directly affected by 30% of the modules and indirectly by 50% (see Figure 1), the recommended teaching time is relatively low, i.e. only 10 hours compared to PRO (33 hours) and SEC (20 hours). There are 40% and 70% direct and indirect connections with the modules for PRO compared with 20% and 0% direct and indirect connections for SEC.

2. Based on the current literature and trends in the industry, quality in SE modules is an essential and substantial element. Nevertheless, quality as a topic is not independent but is built-in and integrated with all the modules. Therefore, quality should be treated as cutting across the modules in SE [7][8].

ELIMINATING THE GAPS

Industry-academia Gaps

Software Engineer Roles for CICD (CI and CD)

Competition requires a faster delivery of software products to market and increases the pressure to be the first to produce new features. Continuous integration (CI) and continuous deployment (CD) are new trends driven by market demand [9]. These trends affect the ability to provide earlier, continuous delivery of evolving software. These new paradigms signify a different approach to the software development lifecycles (SDLC), and the need to understand the impact of this change on software quality and testing procedures. This in turn has implications for the role and responsibilities of the software engineer. These continuous practices enable fast software releases once the deliverables meet the quality criteria, based on continuous testing and validation processes. One of the CICD key enablers is automation support in the development pipeline. Test automation was found essential for removing roadblocks to continuous practices [10].

However, there is no standard implementation pipeline. It is necessary to develop innovative approaches and tools that not only enable team members to receive, build, and test results correctly and in a timely way, but should also be aligned and embedded in the deployment pipeline [10]. Recent studies demonstrated that the lack of knowledge, expertise and skills in Agile methodologies is one of the challenges to be overcome (Agile development is an umbrella term for several iterative and incremental software development methodologies) [2][10-13]. The significant gap in the required skills when implementing continuous practice stems from the need for new technical qualifications and soft skills (e.g. communication and co-ordination). Universities still have curricular units too focused on traditional development paradigms, with relatively little attention to soft skills [11].

Feedback Cycle - *an Important Mechanism*

The gap between academia and industry results from separate learning environments with different goals and motivations. Academia’s objectives are to qualify the future software engineers and to promote research in SE. On the other hand, the main objective of industry is to maximise profits. However, these two worlds have interfaces, dependencies and common goals. Figure 2 demonstrates a possible feedback cycle between industry and academia [14].

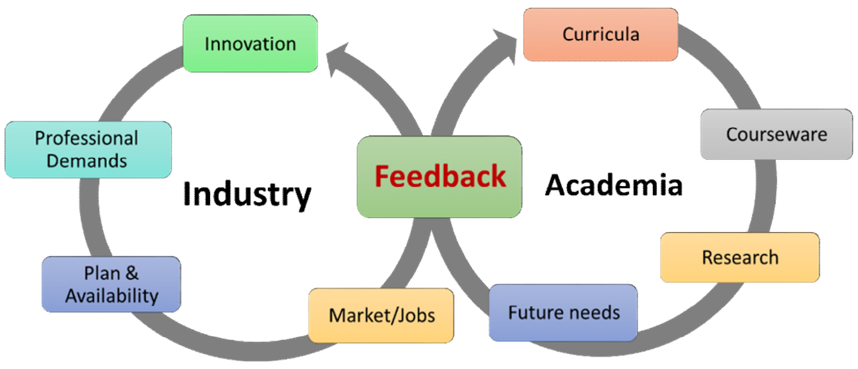

Figure 2: Academia-industry feedback cycle.

Quality and Testing in SE

The authors argue that the quality dimension should be considered in each SE course, from day one of studying. In order to release software continuously and automatically, a solid foundation of soft and technical skills are needed in a more comprehensive approach. It has become clear that the current portion of quality related topics for undergraduate students in not sufficient and does not align with future trends.

Educating students to think *quality* is complex and is intertwined with fundamental and advanced topics in SE throughout the development lifecycle: starting from the meticulous process of requirements engineering, where the customer is a full partner, and continuing with design and continuous testing. The testing layer in this context is based on automation and application programming interface (API) testing rather than traditional manual testing.

The process of continuous testing can be divided into various modules, such as development, continuous integration, quality analysis and performance of the application. These four domains need to be tested in their own unique ways, so that complete end-to-end testing is achieved.

A software engineer task, such as writing code is not an isolated activity, but is part of the quality assurance (QA) and version control processes. Today, the software engineer is required to learn new tools and to be in line with Agile and DevOps methodologies.

Designated (QDev) Unit for Quality and Advanced Testing

The new trends in software engineering raise new challenges that should be studied further. There is a need to adapt the SE curricula to contemporary needs and to further study related topics. For example, the rapid development process is a trigger for the development of appropriate methodologies; smooth and quick delivery requires high quality software. Hence, the role and skills of the software engineer need to change and adapt.

However, dealing with it is not trivial due to prejudice and sometimes the low prestige associated with the quality aspects of the traditional environment. The QA might be perceived as a less-demanding and less-glamorous task compared with programming. The trap here is that good and clear code cannot be achieved without attention to high quality throughout the development lifecycle. The future software engineer should realise this dependency.

The authors founded a dedicated unit for the development of software quality as part of the Software Engineering Department in the Shamoon College of Engineering, Israel.

.

Description of the QDev Unit

The Unit was tailored for contemporary and future needs. The goal of the Unit is to train professional SEs to have a better understanding of quality concepts and their effects on delivered products. Also included are effective management, minimising risk and fostering soft skills.

The quality approach and values should be applied from day one and built into the SE programme. The large number and variety of quality processes along the software development lifecycle require the software engineer to think how to produce quality software as part of the whole project.

Areas of activity include enriching the fundamental courses in the general programme and exposing the students to quality topics, such as test-driven development (TDD), unit testing, automation testing, continuous testing and practices, standards for good writing and the development of soft skills. In addition to participation in the fundamental courses, the Unit develops and offers elective courses, such as:

- Advanced Methods for Software Development;
- Automation Testing;
- Knowledge Management in Advanced Environments;
- Advanced Topics in Software Engineering;
- Organisational Processes in the IT World;
- Security Testing;
- Requirements Engineering.

Facing the pedagogical challenges of SE education, the authors adopted methods that put the student in the centre of the learning [6], emphasising what and how students learn rather than what is being taught. This is also known as *active learning*, in which students are required to do meaningful learning themselves, to think what they are doing during the learning process and to solve problems by themselves rather than passively receiving information from the instructor. Two well-known instructional strategies that adhere to this approach are problem-based and project-based learning.

The similarities of these two pedagogical methods are that they are both based on self-direction and collaboration, and both require feedback and have a multidisciplinary orientation. However, the project-based approach is broader and more sustained. Therefore, the authors chose to implement it as part of the teaching methodology.

The authors believe that adopting this approach may also foster soft skills and other skills, including creativity, self-learning, problem-solving, communication, collaboration and best practices. Therefore, the Unit develops and introduces courses using the project-based approach.

Symbiosis with Industry

The Unit has come to fruition through joint efforts of industry and academia. Academia can provide expertise about the creation and maintenance of the curricula; industry can bring expertise about practices and real-world needs, as well as opportunities for research.

SUMMARY AND CONCLUSIONS

Recent software development trends emphasise the importance and the necessity of collaboration between academia and industry in SE. Nevertheless, the level of industry-academia collaboration in software engineering in general, and in software testing in particular, is quite low [15].

The authors facilitated a quality and testing (QDev) Unit that offers a variety of courses and services to the Department's students and faculty members. The unit's activities reflect the software quality and values that should be applied from day one, from fundamental courses in the first-degree programme to the elective courses in the final projects.

In addition, in order to strengthen the connections and collaboration with industry and other academic institutions, the unit is responsible for an academia-industry conference and roundtable that provides a forum for discussion, as well as an open channel between academia, practitioners and leaders from industry. The authors believe that this is the right way to improve the qualification process and to achieve the best solutions to address current and future problems in software methodologies and software quality.

REFERENCES

1. Spinellis, D., Research Priorities in the area of Software Technologies. EU DG Communications Networks (2016).
2. Fitzgerald, B. and Stol, K.J., Continuous software engineering: a roadmap and agenda. *J. of Syst. Software*, 123, 176-189 (2017).
3. Rowley, J., The wisdom hierarchy: representations of the DIKW hierarchy. *J. of Infor. Sci*., 33, **2**, 163-180 (2007).
4. Sommerville, I., *Software Engineering*. New York: Addison-Wesley (2010).
5. Fowler, M. and Foemmel, M., Continuous Integration (2018), 1 May 2018, http://www.thoughtworks.com/ continuous-integration
6. Ardis, M., Budgen, D., Hislop, G.W., Offutt, J., Sebern, M. and Visser, W., SE2014: curriculum guidelines for undergraduate degree programs in software engineering. *Computer*, 48, **11**, 106-109 (2015).
7. Bourque, P. and Fairley, R.E., *Guide to the Software Engineering Body of Knowledge* (SWEBOK (R)): Version 3.0., IEEE Computer Society Press (2014).
8. Bunyakiati, P. and Phipathananunth, C., Checking software engineering curricula with respect to SE2014 curriculum guidelines. *Proc. 2017 Inter. Conf. on Manage. Engng., Software Engng. and Service Sciences*, New York: ACM, 44-48 (2017).
9. Hilton, M., Understanding and improving continuous integration. *Proc. 2016 24th ACM SIGSOFT Inter. Symp. on Foundations of Software Engng*, New York: ACM, 1066-1067 (2016).
10. Shahin, M., Babar, M.A. and Zhu, L., Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5, 3909-3943 (2017).
11. Almeida, F., Challenges in migration from waterfall to agile environments. *World*, 5, **3**, 39-49 (2017).
12. Chen, L. Continuous delivery: huge benefits, but challenges too. *IEEE Software*, 32, **2**, 50-54 (2015).
13. Leppänen, M., Mäkinen, S., Pagels, M., Eloranta, V.P., Itkonen, J., Mäntylä, M.V. and Männistö, T., The highways and country roads to continuous deployment. *IEEE Software*, 32, **2**, 64-72 (2015).
14. Kenett, R.S., Ruggeri, F. and Faltin, F., Analytic Methods in Systems and Software. Wiley (2018), 1 May 2018, https://www.wiley.com/en-au/Analytic+Methods+in+Systems+and+Software+Testing-p-9781119271505
15. Garousi, V., Felderer, M., Kuhrmann, M. and Herkiloğlu, K., What industry wants from academia in software testing: hearing practitioners' opinions. *Proc. 21st Inter. Conf. on Evaluation and Assessment in Software Engng.*, New York: ACM, 65-69 (2017).