Preparing software engineering students for industry

Vreda Pieterse, Fritz Solms and Stacey Omeleze

A software engineering course should not only provide students with a theoretical knowledge of software engineering practices and tools, but also equip them to apply these in the industry. Creating multiple opportunities for students to develop a range of employability skills is therefore an essential component of a software engineering course that seeks to produce industry-ready graduates.

The Department of Computer Science in the Faculty of Engineering, Build Environment and Information Technology focuses on teaching its software engineering students both the technical and workplace skills required to thrive in the software engineering industry. Its third-year software engineering module focuses on teaching students the essentials of software engineering processes and their complexities, exposes them to a variety of methodologies for tackling different stages of the software life cycle, creates awareness of the latest trends in software engineering and support tools to assist in the implementation and control of projects under development, and facilitates the execution of a team project in order to introduce students to the responsibilities associated with a variety of roles in a teamwork scenario. The motivation behind this approach is that students will benefit largely from the practical skills that they have developed when entering the workplace.

To succeed in its goal, the software engineering module partners with industry role-players to give students the opportunity to experience the intricacies of working on a real project. During the first eight to ten weeks of the module, the students work on a class project consisting of a series of micro-projects in which different teams work collaboratively to design and implement a modular system. The students are assigned to new teams for each micro-project to create maximal opportunities for them to develop their employability skills of communication, management and planning, teamwork and collaboration, interpersonal relations and problem solving. To avoid free-riding, much effort is devoted to ensuring that the lecturers in the module understand the working patterns of each team, as well as the individual team members, throughout the semester’s projects. This helps to ensure that each member of the team is prepared for the challenges that await them in the work environment.

Following the class project, the remainder of the module is dedicated to industry-based capstone projects, where students participate in self-selected teams of five members. Each team has to develop a large, authentic, open-ended software system for a client in industry. They are required to liaise with this client and present progress reports of their work at regular intervals. The main goal of this project is to deliver a high-quality, cost-effective software project to the client, while simultaneously mastering technical software engineering skills and applying cutting-edge technologies to a practical problem.

The technical skills that students enrolled for the software engineering module need to master through their projects include the following:

Configuration management and issue tracking

Students are required to manage documents and code that are collectively owned by the members of the team through configuration management tools that track code changes, and to facilitate a team of people to work concurrently on the same artefacts, resolving conflicts as needed. These tools support base-lining, rolling back to previous versions and branching. Branching enables developers to develop new features and fix bugs in a separate branch that is merged back into the main trunk only after having passed both unit and integration testing.
Buisman and van Eekelen (2014) observe that students are reluctant to perform atomic commits with short clear commit messages and to create branches rather than pushing to master. This degrades the quality of the work and diminishes the depth of the learning that takes place. For this reason, the lecturers of the software engineering module evaluate how the students use their configuration management tools on a regular basis. When forced to comply with the criteria for the proper use of these tools, students get acquainted with them and gradually grow into appreciating their value.

**Unit and integration testing**

The module emphasises the need for formal software testing throughout the development process. Through unit testing, the students learn that, although the different modules developed by the different teams are dependent upon one another, each sub-system can be tested in isolation. Thereafter, integration testing ensures that each component works within its actual deployment environment.

**Component-based design and decoupling**

The module requires technology-neutral component-based modelling, facilitating the mapping of component contract specifications into interfaces and unit tests, as well as data structure specifications into entities and activity diagrams into method bodies. This introduces students to component-based development based on interface control diagrams, but does not require component contract specifications and full decoupling via dependency injection. In contracts-based developing, one codes against contracts for the dependencies and not against the concrete components. Contracts include the specification of the required functionality with the pre- and post-conditions for each method, service or function. Dependency injection is an implementation of the Inversion of Control (IoC) pattern, which is widely used in industry to achieve loose coupling by having concrete components for required dependencies injected into the dependent component. This facilitates unit testing against mock objects, integration testing against actual components, as well as deployment into different environments where different concrete components may be available to provide the required functionality. Modern reference architectures and frameworks like Java-EE, Spring, Django, node-js and Microsoft.Net all provide IoC containers to implement this software engineering technique.

**Build tools and integration**

Modern software systems are commonly complex, and require sourcing dependencies, configuration, compiling, testing, linking and packaging, distribution and the deployment of a variety of components, which may themselves be developed in different technologies. It is standard practice in industry to use platform-independent build tools that automate this process. There are many examples of these tools. The integration of such modules into a cohesive system is often subject to a range of unforeseen complexities and challenges, and students must learn to deal with these.

Because the capstone projects are sourced from industry partners, their authenticity increases. This typically increases the student’s motivation. It also enables the Department to monitor changing trends and technologies in the industry, as well as assess any gaps between the skills developed through the software engineering course and those required by the industry.

Industry projects are vetted for appropriate scope and complexity, as well as the potential to capture the interest of the students before they are presented to the industry client. The teams are then required to submit tenders for the three projects in which they are most interested. Industry clients express their preference for the teams based on the strength of their tenders, after which partnerships are established. The teaching team assigns projects to teams giving first preference to the choice of client and second preference to the choice of team, thereby rewarding teams that formulated strong tenders.

Through this process, industry clients become educational partners in the training of the students and receive the opportunity to work with potential future employees.

The course culminates in an end-of-year project exhibition, through which the students have to contend with the prospect that their efforts will be laid bare for public exposure, as would be the case in a real-world scenario. The exhibits are judged by staff members, as well as industry representatives, who contribute towards prizes in various categories. These can include innovation, software engineering excellence, algorithmic innovation, architectural awareness, user experience, and triple bottom line.

**References**