Teaching software systems thinking at The Open University

Teaching Software Systems Thinking at The Open University

Michel Wermelinger, Jon G. Hall, Lucia Rapanotti, Leonor Barroca, Magnus Ramage, Arosha Bandara
Computing and Communications Department, The Open University
Walton Hall, Milton Keynes MK7 6AA, UK

***Abstract*—The Open University is a distance-based higher education institution. Most of our students are in employment and study from home, contacting their tutor and fellow students via e-mail and discussion forums. In this paper, we describe our undergraduate and postgraduate modules in the software systems area, how we teach them at a distance, and our focus on shifting our students' minds into a reflective, critical, holistic socio-technical view of software systems that is relevant to their particular professional contexts.**

I. Introduction

The Open University (OU) is the UK's largest university, with ca. 200,000 students. Apart for our on-campus PhD students, study with the OU is at a distance: students are assigned a tutor and provided with printed and online learning materials which they study at their leisure, towards fixed-date assessment points, in their own social and professional context. Most of the students on our computing programmes are employed, with most of our postgraduate students already working in the IT industry. Tutors mark their students' assignments, answer their queries, and organise tutorials, mostly online. Tutors in the computing programmes are largely practicing or retired IT professionals or academics at other universities.

The 2010 Independent Review of Higher Education Funding and Student Finance (aka Browne Review[1]) increased indirect pressure on UK universities to consider their output differently: with course fees increasing substantially, students now see degrees as commercial products and have expectations concomitant with that status. To pay fees, most students ask for government loans, which are only available to students registering for a degree. This had a major impact on the OU: our modules, whilst embedded in degrees, were designed to be as much as possible self-contained so that students could pick and mix modules to suit their personal interests or professional needs, without necessarily taking a full degree.

These, and other drivers, have caused a re-evaluation of the whole OU computing curriculum at both undergraduate and postgraduate levels. The results of that re-evaluation—new modules in new structures—are now being offered to students.

A. The undergraduate context

In 2011, we carried out a major review of the level 3 (final year) computing curriculum, against a background of a changing landscape, where desktop computing is being overtaken by ubiquitous computing, with the web becoming a universal utility, and where outsourcing and globalisation mean that technical skills need to be at a higher level.

The main outcome of that review was that our graduates would need to understand the issues surrounding:

- people and systems everywhere – from how different groups of people will work and use software systems to how people develop, design and maintain these systems;
- information services everywhere – from modelling and storing information to accessing services on mobile platforms and the cloud;
- devices everywhere – from fixed and mobile networks to the experience of living with and building for the Internet of Things.

Three clusters of modules were planned to address those three issues, with the people and systems cluster being the first developed, and the other two currently in production.

The focus of the people and systems cluster goes beyond the initial design of the system to its use and maintenance, for which issues of organisational context and change are crucial. It is principally rooted in software engineering and information systems, but it also draws upon systems thinking, management (including project management), and science and technology studies. This cluster was divided in two modules, further described in Section II.

B. The postgraduate context

Our previous postgraduate curriculum [1] was based on 15-credit modules and a 60-credit dissertation (research project). This range of small modules allowed students to configure their study to suit their interests and needs. However, it led to many assignments and exams to get a Diploma or MSc qualification, making them less attractive. This, and the OU's move to a qualification-focussed curriculum, led to a new postgraduate curriculum with 30-credit modules and explicit specialisations in:

- Information Security and Forensics, with compulsory modules *Information Security* (M811) and *Digital Forensics* (M812);
- Software Engineering, with compulsory modules *Software Development* (M813) and *Software Engineering* (M814), which together span the body of knowledge specified in the SWEBOK [2].

The MSc research project must be aligned with the specialisation. Section III describes the new modules except M812, which is outside the scope of this paper.

[1] http://en.wikipedia.org/wiki/Browne_Review

II. The undergraduate modules

A. TM353 — IT Systems: Planning for Success

This module draws heavily on previous postgraduate teaching in information systems. It starts from the observation that, however well designed technically, IT systems in practice fail very frequently. The premise of TM353 is that this is due to the socio-technical nature of IT systems in use: that they are a complex mixture of technology (including hardware, software and networking), organisations, and people.

The goal of TM353 is to equip students with skills to enable them to plan, design and implement IT systems which are successful in use. The concept of socio-technical systems design is not new — it goes back to the work of the Tavistock Institute on industrial systems such as coal mining and manufacturing in the 1950s, and has been applied to information technology since at least the 1970s [3]. In recent years, it has been explicitly linked in some literature to software engineering [4][5], the latter calling for a new field of "socio-technical systems engineering". However, a concept of socio-technical systems is frequently absent from computing curricula.

This module is strongly based within systems thinking. The OU has a 40 year history of teaching systems thinking, making considerable use of diagrams as a form of qualitative modelling as this has proved accessible to distance education [6]. A particularly relevant form of systems thinking that has arisen at the OU is the systems failure method, which identifies the systemic causes of failures and enables organisations to learn from them for the future [7]. The OU systems traditions are enhanced by techniques from socio-technical systems design, especially the work of Mumford [3]; and also the considerable recent work on complexity theory and the complex interactions between components in a large-scale IT system.

In addition to the systems framework that forms the basis of the module, a series of concepts and techniques are drawn in as appropriate from other areas related to IT systems. These are concerned either with initial design to ensure systems success (including issues of power and stakeholder analysis, information systems methodologies, security and privacy, reliability and dependability), or with actions to ensure ongoing success (including information systems evolution, scenario planning and disaster recovery). Running throughout the module is a strand concerned with legal, social, ethical and professional issues, and with project management.

The nature of IT systems success is not taken for granted. The module acknowledges that this is a contested question, deeply bound up with different stakeholder perspectives and the power relations between stakeholders: what is a success to one group may look quite different to others. The nature of failure, and the different reasons why systems fail, is also examined in detail, and a distinction drawn between success in the execution of a project to develop and implement an IT system, and its success at effectively meeting the needs of the organisation where the system is situated.

Although TM353 introduces a significant number of practical tools and concepts, the module is ultimately aiming at a mental shift among students: the development of an awareness that IT systems are inherently socio-technical, and that their success or failure arises from socio-technical factors rather than purely technical ones.

B. TM354 — Software Engineering

This module teaches the principles, patterns, techniques and practice associated with requirements engineering, analysis, software architecture and design, as well as the principles and techniques of implementing and testing a software system. Students get a sound understanding of the quality issues involved in software products and processes. The module follows a plan-driven development contrasting it at each stage with an agile approach; for example, while studying requirements documentation using the Volere template[2], the students also listen to, and write about, an interview[3] where the authors of the template discuss their perspective on the implications of an agile approach to requirements and their documentation. Students develop a practical awareness of different approaches to software development and an understanding of agile practices.

Although modelling techniques are taught, and a notation like UML is used throughout the module, the emphasis is not on the details of the techniques but rather on understanding a problem and its context, to help students make their own judgements on what is most appropriate for a specific situation.

Students are encouraged to be creative and to work with others, sharing their artefacts and reflecting on feedback from colleagues. They use a collaborative tool where they upload models and give comments to, and receive feedback from, colleagues. As part of their assessment they need to reflect on changes they make to their own artefacts as a result of comments received. In a distance teaching setting, this is a way to simulate what would happen in an agile stand-up meeting in the process of reaching a shared understanding of a problem/solution.

TM354 was designed with the preoccupation to develop critical reflection. As a level 3 module, it also prepares students for the final project in software engineering, developing the required skills, e.g., conducting research searches, assessing and reviewing found material. To promote an understanding of the professional context, TM354 brings in regularly updated topics on software engineering practice, challenging students to confront what they learn with what practitioners do.

III. The postgraduate modules

A. M811 — Information Security

Whether in the public or private sector, it is the value invested in the information assets of a modern organisation that underpins its effectiveness and drives its profitability. M811 explores the professional and technical skills necessary to understand, document, manage and implement strategic and operational aspects of an organisation's information security.

[2]http://www.volere.co.uk/template.htm

[3]http://www.se-radio.net/2012/09/episode-188-requirements-in-agile-projects/

M811 teaches important and transferrable topics in information security risk assessment and management, as well as professionalism, home information security, and information security research.

Every organisation in the UK has Information Security responsibilities. Many large organisations (but by no means all!) do have hard InfoSec stances, understanding the criticality of their information assets, their threat landscape, and the relationship of policy, technologies and strategy in protecting them. Other organisations, typically smaller ones, have a weaker understanding of their InfoSec needs. They are, however, constrained by the availability of InfoSec knowledge that is directly applicable to their InfoSec needs. M811 is aimed at employees of such organisations and, as we will describe below, provides a fit-for-purpose pilot InfoSec management system for that organisation.

Students are located throughout their study within their own organisation. This provides a very rich context for the module to draw from and permits the student to deliver value back into their organisation from their study.

The role of the student on M811 develops from learner at module start to InfoSec problem solver at module end. The problem solving skills taught align with the international InfoSec standard and are based on industry standard texts. What M811 adds is a detailed conceptual framework that underpins leading edge professional skills, giving the student their own updatable learning framework which includes the research literature and the incredibly rich, essentially daily updated resource that includes blogs and podcasts. We achieve this by encouraging the student to explore the InfoSec world and to learn where are the valuable resources that are relevant for their and their organisation's professional situation.

All assessment in the module is based within the student's organisation. Typically, a student will identify a problem with their organisation and bring it into the module. Extending the module's conceptual basis is a framework for guided reflection. Almost 50% of the assessment is the student's own reflection on their organisation's problems and their candidate solutions for them. We provide a generic marking scheme to the tutors to allow them to assess their students' work consistently, despite the diversity of work to assess.

Using the student's rich context has benefits for M811 in that it uniquely contextualises the module's materials for the student, forcing them to deal with real-world complexity [8]. Typically, traditional assessment needs to provide time-boxed mechanisms for the validation of the student's work; questions from this year cannot be reused next year as the answers will be the same. As the student's rich context provides a unique assessment environment, M811 assessment materials do not have to change year on year.

B. M813 — Software Development

M813 is the first module in the new postgraduate Software Engineering specialism. It teaches a wide range of software engineering theories, principles and techniques across the life cycle, with particular emphasis on problem definition, analysis, design, implementation and testing, and includes systematic and creative application to practice through a wide range of transferrable, professional and research skills, including critical evaluation.

With a similar approach to M811, M813 takes advantage of its students' rich professional context to close the gap between academic learning and professional practice, an acknowledged major challenge in software engineering education [9], [10], [11]. Via the module's assessment students engage with a development problem of their choice, working towards a software system for an organisation they are familiar with, and interacting with stakeholders in that organisation. M813 uses the same generic marking scheme approach as M811.

As part of their submission, alongside software development artefacts, which demonstrate to which extent they have mastered specific approaches and techniques, students are also required to provide a written commentary in the form of guided reflection, including both an articulation of justifications and rationale for their choices in the application of what we teach within their real-world context and the lessons learnt in the process, as well as a critical reflection on the teaching as it stands against their own practice. The latter closes a feedback loop which is particularly useful to us as academics as it enables the evaluation of what we teach and its relevance and appropriateness in current professional practice.

C. M814 — Software Engineering

Continuing from M813, M814 is aimed at students interested in developing their systems analysis and management skills, e.g., software developers taking on system analyst or project manager duties, or managers coming from other disciplines. The module examines software's role in organizations from human, social, knowledge, business, and domain problem (requirements) perspectives [12], thereby providing a system perspective to students.

On the one hand, M814 provides an in-depth exploration of the requirements engineering process, starting from stakeholders, goals and scope of software projects to elicitation, analysis and communication of requirements, centering on the Volere approach, as described in the set book [13].

On the other hand, it covers topics like ethical issues, intellectual property rights (including an interview with a patent examiner at the European Patent Office), human motivation, risk assessment, quality and knowledge management, and software evolution, many of which are required for professional accreditation, e.g., by BCS, The Chartered Institute for IT.

In order to make students reflect holistically about the multiple relationships and mutual influences between software and the wider organisational context, some assessment questions ask students to analyse a fictitious case study from multiple perspectives: human resource management, ethical and legal issues, choice of development process, etc. Those and other questions often have no clear right or wrong answer. Instead, the marking scheme rewards students for the arguments put forward, for the breadth of their knowledge, and for 'seeing both sides of the coin'.

Information literacy and critical analysis skills are also key learning outcomes for M814. Students have to critically read literature to sift the original from the incremental and the evidence from the hype, a valuable professional skill in the fast paced IT industry. Criteria to evaluate papers are given in the Course Guide. Each year, we strive to include one academic paper and one magazine article to expose students to different types of literature. Such papers help M814 maintain currency and provide complementing or contrasting perspectives on the issues covered in the M814 text.

A precursor module helped students gain first hand experience of managing the variety of stakeholder viewpoints that must be reconciled as part of the requirements engineering process, through a collaborative requirements elicitation and analysis exercise based on a case study scenario. Despite the challenges of undertaking this activity asynchronously, at a distance, many students valued the experience because it helped them gain a better understanding of the requirements engineering process. We have retained this collaborative working activity in M814, modifying it to use an industrial configuration management tool (Git on GitHub) to collaboratively edit the requirements specification documents.

M814 uses 3 dynamic system models defined with Vensim and custom-made interfaces developed with Sable[4]. The models illustrate trade-offs between productivity, cost and quality [14], Brooks' Law, and Lehman's 2nd Law (as a system evolves its complexity increases, unless work is done to maintain or reduce it). Each model provides some parameters that students can change (e.g., the team size for Brooks' Law) before running the simulation and seeing the results of output values (e.g., time to complete the project). Each model comes with various activities for students to become familiar with the model. Solutions to activities are provided to tutors, but not to students, so that they can discuss the activities in the online forums. The assignments always include a question about one of the system models, for example asking students to find input parameter values that lead to a particular scenario and to reflect on the use of such simulations for management decisions.

IV. Concluding Remarks

In this paper we have described the new postgraduate and final year undergraduate software systems offering at The Open University. Changes in the IT and UK higher education landscapes have led to a new curriculum in which relevant technical and soft skills are blended to provide a rounded academic education to professionals. Topics are drawn from multiple disciplines so that students appreciate, from ethical, economical, management, and other perspectives, the rich socio-technical systems formed by software, its stakeholders and their organisations, and apply systems thinking to ascertain risks, solve problems, and prepare for systems failure.

The modules' content and assessment are carefully designed so that they can be delivered at a distance and at scale to students who have to fit study into often busy personal and professional lives. For example, the type and workload of technology-mediated collaborative activities is considered.

Although the undergraduate and postgraduate modules described cover similar ground, the latter have more open-ended, reflective, and contextualised assessment. Higher education delivers value to the student through validation of their learning. Locating that education in the student's rich professional context enhances and extends the value proposition for the student, who is then able to contribute to the solution of system-wide problems faced therein.

For M811 and M813, we have begun evaluating the novel situated approach to teaching, including their impact on students' learning and how they are received. We will report preliminary outcomes soon. Here we have provided the rationale for the revised programme and the concepts of the modules taught there, informed by several decades of experience at The Open University in crafting high-quality distance learning.

[4]http://www.ventanasystems.co.uk/services/software

References

[1] B. Quinn, L. Barroca, B. Nuseibeh, J. Fernandez-Ramil, L. Rapanotti, P. Thomas, and M. Wermelinger, "Learning software engineering at a distance," *IEEE Software*, vol. 23, no. 6, pp. 36–43, 2006. [Online]. Available: http://oro.open.ac.uk/12504/

[2] P. Bourque and R. Failey, Eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*. IEEE, 2014. [Online]. Available: http://swebok.org

[3] E. Mumford, "The story of socio-technical design: reflections on its successes, failures and potential," *Information Systems Journal*, vol. 16, pp. 317–342, 2006.

[4] J. G. Hall and L. Rapanotti, "Problem Frames for Socio-Technical Systems," in *Requirements Engineering for Socio-Technical Systems*, A. Silva and J. L. Maté, Eds. Idea Publishing Group, 2005, ch. 19, pp. 318–339.

[5] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with Computers*, vol. 23, pp. 4–17, 2011.

[6] M. Ramage and K. Shipp, "Expanding the concept of model: the transfer from technological to human domains within systems thinking," in *Ways of Thinking, Ways of Seeing*, C. Bissell and C. Dillon, Eds. Springer, 2012, pp. 121–144.

[7] J. Fortune and G. Peters, *Information systems: achieving success by avoiding failure*. John Wiley, 2005.

[8] J. G. Hall and L. Rapanotti, "Masters-level Software Engineering education and the enriched student context," in *Proceedings of Joint Software Engineering Education and Training Workshop (JSEET)*, 2015.

[9] L. Brodie, H. Zhou, and A. Gibbons, "Steps in developing an advanced software engineering course using problem based learning," *Engineering education*, vol. 3, no. 1, pp. 2–12, 2008.

[10] L. Johns-Boast and S. Flint, "Simulating industry: An innovative software engineering capstone design course," in *Frontiers in Education Conference*. IEEE, 2013, pp. 1782–1788.

[11] I. Richardson, L. Reid, S. Seidman, B. Pattinson, and Y. Delaney, "Educating software engineers of the future: Software quality research through problem-based learning," in *Proc. 24th Software Engineering Education and Training Conf.* IEEE, 2011, pp. 91–100.

[12] P. Hall and J. F. Ramil, *Managing the Software Enterprise: Software Engineering and Information Systems in Context*. Cengage Learning, 2007. [Online]. Available: http://edu.cengage.co.uk/catalogue/product.aspx?isbn=1844803546

[13] S. Robertson and J. Robertson, *Mastering the requirements process*, 3rd ed. Addison-Wesley, 2006.

[14] D. Pfahl, M. Klemm, and G. Ruhe, "A CBT module with integrated simulation component for software project management education and training," *The Journal of Systems and Software*, vol. 59, pp. 283–298, 2001.