Teaching web technologies: understanding the tutor's perspective

Teaching web technologies: understanding the tutor's perspective

This paper describes an Open University eSTEeM project that gathered the experiences of computing and information technology tutors who teach an undergraduate module called *Web Technologies* with the intention of understanding more about how they and their students can be best supported. Twelve distance learning tutors were interviewed by two interviewers. The interviews were transcribed and then thematically analysed. It was discovered that some tutors hold the view that some of their students struggle to understand aspects of the technologies that are being taught. It was also apparent that tutors have very different approaches when using certain tools to teach web technologies to students. The paper concludes by summarising key findings, presenting potential enhancements and suggesting further research directions.

Keywords: pedagogy, computer programming, web technologies, qualitative study, thematic analysis

Introduction

Web technologies is a second level module, which is equivalent to a second year module in a full-time undergraduate degree, taken by students who are studying towards a computing and IT qualification from The Open University. As its title suggests, the module introduces a range of important software technologies. This article presents qualitative research that has been carried out to uncover the experience of tutoring a technologically demanding computing module.

The article begins with a description of the module. This is followed by a summary of the research questions, a discussion of the data collection and analysis methods that were adopted. The next sections provide a summary of the data and an accompanying discussion. Towards the end of the paper a set of conclusions are presented.

Web Technologies

The Open University *Web Technologies* module, which is also known by its module code TT284, is a second level computing and IT module that introduces students to a range of software technologies that are essential to the operation of today's web-based systems and products.

The module comprises four blocks (or units of study): *Foundations of web technology*, *Web architectures*, *Mobile content* and *Developing applications*. All blocks are supported by a series of complementary case studies. The first block introduces students to 'basic client server architecture; protocols such as HTTP; content markup (HTML, CSS, XML) and issues of accessibility and usability' (The Open University, 2018). The second block has a more practical focus: students are required to adapt and then implement a simple web application that can be run from a desktop computer, using technologies such as JavaScript, SQL and PHP. The third block requires students to create a simple mobile app. The final block enables students to gain knowledge and understanding about how to plan and manage the development and deployment of web projects.

The version of the module that was the focus of this study included three sports-related case studies. These case studies show different ways that web technologies can be used and depict web-based applications that are of different sizes. The first sets out the requirements for software needed by the Open University running club. This case study is used to help students create a web-based application to store information about running events. It is complemented by the requirement for students to create a simple stopwatch app that is used on a mobile device, such as a smartphone. The second case study focuses on a 'grass roots' sports initiative called parkrun (parkrun UK, 2018). An important part of the parkrun case study is that it demonstrates technology that enables

participants, who are runners, to record their running times and overall performance. The final case study relates to the web infrastructure of the London 2012 Summer Olympics. This case study was chosen because the Olympics presented some interesting technical challenges: the web infrastructure had to be both robust and scalable to handle large numbers of users wanting to access information at different times during the event.

Module assessment is through three tutor-marked assignments (TMAs) and an end-of-module assessment (EMA), which is akin to a final examination. The TMAs are an important element of the Open University's method of distance teaching: they enable students to demonstrate their understanding of key concepts in the module, and present opportunities for their personal tutors (who are known as associate lecturers) to offer constructive and helpful feedback on their progress.

The first assignment requires students to show their understanding of HTML and accessibility by making some sample files standards compliant and by writing a short report about the accessibility of a prototype of the Open University running club website.

The second assignment is more demanding, but adopts a similar approach: it requires students to edit and enhance example code to add client-side form validation (error checking) and store records to an SQL database. SQL is an abbreviation for Structured Query Language; it is a programming language that enables programmers to create instructions (or queries) to add, retrieve and update database records. In completing this task, students have to grapple with PHP, a server-side language that generates web pages and connects with other software components such as databases (PHP, 2018), and JavaScript, a client-side language that runs in an internet browser. Another part of the assignment required students to reflect on the technologies that they

have used during the module, and suggest alternatives. This writing component exposes students to other ways of solving the same problems with different technologies. The third assignment is concerned with mobile technologies: students are required to create a prototype app that relates to the running club case study.

Students are likely to have studied two level one modules before enrolling on *Web technologies*. At the time of writing these introductory computing modules were *TU100 My digital life* and *TM129 Technologies in practice*. *Web technologies* is a module that is about computer programming. This said, it does not explicitly teach programming since students are expected to already understand the principles of problem decomposition and the key elements of imperative programming languages, such as programming constructs, functions and variables.

Research objectives

Our overarching research objective of the project was to listen to the tutor's voice to learn more about the challenges of teaching *Web technologies*. This objective led to the creation of a number of connected research questions. The first was whether tutors believe that their students are adequately prepared for studying a second level computing and IT module such as *Web technologies*. This question was especially important since both *TU100 My digital life* and *TM129 Technologies in practice* use visual programming languages. *TU100 My digital life* introduces students to a programming language called Sense, a dialect of Scratch (MIT, 2016). The Sense language introduces students to key programming concepts using graphical building blocks that can be combined together to solve simple problems. *TM129 Technologies in practice* introduces students to a language that controls a simulated robot. The *Web technologies* module, on the other hand, makes use of popular textual languages, such as JavaScript and PHP.

A further reason for studying the programming aspect of the module was that some students may struggle with this very important aspect of computing. In a review of the teaching and learning of programming Robins et al. (2003) say that "novice programmers face a very difficult task. Learning to program involves acquiring complex new knowledge and related strategies and practical skills". With these complexities in mind, a further question to ask tutors was: what aspects of the *Web technologies* module do you think students find most difficult? It was believed that answers to this question would be informed by instances where experienced tutors were asked to offer help and guidance to students who struggled with key aspects of their learning.

The final research question asks what tutors believed might be done to improve or enhance the module. This question was considered especially important since the module is presented entirely online: teaching is performed through correspondence tuition, discussion forums (Nandi et al., 2012) and real-time synchronous tutorials through online rooms where students are able to directly speak with their tutor.

Methodology

This was a qualitative research project which used interviews with tutors to learn about their experience of teaching *Web technologies*. At the start of the project a call for 'tutor-collaborators' (co-researchers) was circulated to all tutors who were employed on the module with the intention of recruiting an experienced tutor who would act as a research assistant to conduct and coordinate the tutor interviews. The call for co-researchers yielded a very positive response; many tutors wanted to play a central role in the research project. Faced with a significant number of very impressive applications, two co-researchers were chosen by considering the breadth and depth of their experience and their availability. The extent of their subject knowledge and teaching

experience was considered to be an essential criteria, as experienced tutors should be able to more readily draw on their background to successfully guide and inform their interviews with tutors.

The co-researchers were briefed in terms of the research objectives and were asked to prepare a detailed interview plan (Appendix 1). To prepare for the interviews, and to test the interview plan, each tutor-collaborator took turns to interview each other about their experience of tutoring on the module. In doing so, they were able to identify areas of the plan that needed updating or modification and, secondly, it enabled both co-researchers to gain experience and familiarity with using the interview plan.

A total of 12 tutors for interview were randomly selected from a list of 27. Each of the 12 tutors were asked whether they would like to participate in the project. Despite not being offered an incentive, all 12 tutors agreed: 6 were interviewed by one tutor-collaborator, 6 by the other.

Before each interview, each participant was told that all interviews would be anonymised and they were free to stop the interview at any point. Participants were also told that all interviews would be recorded and then transcribed. Since all participants were tutors who worked remotely, it was decided that they would be interviewed through OU Live, the university's online tutorial tool that was used at the time of the study. The advantage of this approach is that OU Live could be used to share a summary of the interview questions and to record an entire interview. After each interview was completed, the interview exported to an audio (MP3) file, which was then emailed to a transcriber.

When all the interviews had been completed the interview transcripts were read and then loaded into a qualitative analysis software, NVivo, for thematic analysis (QSR, 2016). The analysis was an iterative process that roughly adopted the phases suggested

by Marshall and Rossman (1999): categorisation, coding, testing of understandings using the data, and searching for alternative explanations.

An important aspect of this type of research is the background and perspective of the researcher who is interpreting the data. This is in keeping with the view that '[a]ll social research is founded on the human capacity for participant observation' (Hammersley and Atkinson, 1995, p. 18). It is important to state that the author has worked as an Open University tutor for approximately ten years. His main role within the university is to line-manage a group of Open University tutors (associate lecturers); he communicates with them on a daily basis and works with them to support their students. Before joining The Open University he worked as a professional software developer for over six years using many of the technologies that are featured within the module.

After completing the thematic analysis, a meeting was arranged to share the initial results with the co-researchers. A summary of the key findings were presented along with key themes that had emerged through the data. Since the raw data were available through NVivo, the co-researchers were able to view (and critique) the tags (or 'codes') that had been used to annotate their interviews.

Results

The tutor interviews ranged between 40 minutes and 1.5 hours in length. Each interview was transcribed. The resulting interview corpus was over 110,000 words, and over 250 pages in length. During analysis, 108 unique nodes, or 'discussion themes', emerged from the data. These ranged from topics about the technologies that were being taught through to comments about module tuition philosophy. These topics were used to identify, or tag, 850 fragments of text. Coding took place by closely analysing and re-reading the text, and carefully considering existing themes to constantly determine

whether new themes needed to be introduced.

The top ten most referenced themes were as follows: forums (62), OU Live (62), JavaScript (44), challenges (35), PHP (34), student background (29), module structure (28), struggling students (27), tutor background (25) and pedagogy (25). These broad themes have a direct relationship to the topics that were presented to participants through the interview questions. The data also contains 37 themes that were only referenced once across all interviews.

It was clear from the interview data that participants were free with their opinions and generous with their time. In fact, since distance education can be a solitary activity, participants seemed to welcome the opportunity to discuss the challenges of tutoring.

Participants emphasised that *Web technologies* attracted a wide range of students. Some students may begin study with a pre-existing knowledge of some of the key technologies that are featured in the module, such as PHP and JavaScript, whereas others begin with very little; this wide variation is not uncommon within The Open University due to its open access policy.

An interesting and important theme was that participants reported that some students struggle with programming. In particular, they voiced concerns about debugging saying, for instance, that ‘debugging your code is not easy’, ‘perhaps they don’t know how to use debugging, perhaps a session on that might be useful or perhaps given them some hints and tips’ and ‘basic debugging skills; some just don’t seem to be understand how to test and if they do have a problem they just don’t seem to be able to understand how to work out where the problem is’. Another participant reported that ‘the biggest problem with students is that, particularly the weaker students, lack strategies of how to understand why things don’t work’. The same participant went as

far as saying, ‘I actually think that is something that we don’t really teach them; we almost hope that it is a skill that they will pick up; we don’t really teach it to them, and I think that is part of the problem’.

Continuing with the subject of programming, another participant questioned the strength of problem-solving skills: ‘if they don’t know the tools that are available to them in order to be able to solve the problem then how can they possibly solve the problem’. In this context, ‘tools’ are interpreted to mean an understanding of the fundamental structures of programming, and general ways to decompose problems.

In the second block of the module students are required to use JavaScript, regular expressions, PHP and SQL. Participants had different views about the extent to which students struggled with JavaScript, primarily because it used other pieces of technology. One participant explicitly reported that ‘I would probably put JavaScript at the top of the list in terms of difficulty’. Others raised concerns that related to programming knowledge: ‘the trouble is that obviously we are starting to deal with concepts of arrays and I think that this is where the students who are newish to this kind of thing are going to have a problem’. Other points related to debugging, as participants said ‘I think that the problem that they have with JavaScript is … being unable to spot errors and detect errors’ and ‘I will look at it [student code] within a web console and it throws up all these errors and they have obviously never looked at [it]’.

Some participants reported that some of their students find the pedagogic approach adopted by the module challenging. In keeping with a desire to teach professional skills, the module team direct students to publically available tutorial resources and familiar computing text books, the same resources that professional software developers are likely to use. Participants anecdotally reported that some

students struggle to use these resources effectively, citing that some students may feel overwhelmed.

Students are required to use regular expressions to validate data before records are sent to an SQL database for storage. Regular expressions is a well-known pattern matching language found in many different programming languages, including JavaScript. Participants were consistent in their views that regular expressions are difficult for students. They also reported that students ‘certainly seem to struggle the most with regular expressions’ and find that ‘regular expressions are really hard and I don’t think there is any getting away from that, and they are hard for everybody’. Participants hinted at potential solutions. These included providing further explanations or even adopting an alternative approach. One said, ‘to be honest these days … I would use a validation library, or something like that’.

In one assessment students are required to write PHP code that receives data from a web client and save that data to a database. Some participants reported that there was not enough context or background about the PHP language and said, for instance, ‘I think that PHP is grossly undersold in this course. It is presented purely as a database interface and I think it is massively undersold’. Understanding that PHP code runs on a server is an important part of a wider mental model that helps students understand how the different technologies are combined together. One participant reported that ‘quite a few will find the concept of moving onto server type technologies really a head bashing idea and the number of them that try and run PHP locally is quite amazing’.

Participants held opinions about how students are taught SQL. One participant was very clear: ‘I think what the students also need is a set of practice SQL tables, which are nothing to do with the TMA so that they can practice on them, and we could use in tutorials to help [the students to] understand concepts’. The same participant had

a very clear vision of how such a resource could be used: ‘if we had a set of dummy tables then we could use application sharing and say [to our students] look, this is how you connect to a database’. Another participant offered a complementary view: ‘I think it would be quite helpful … to have a few extra practice activities for those that need the practice, and a couple of stretch activities for those that are more familiar [with databases]’.

The other significant programming component of the module, MIT App Inventor (MIT, 2016), exposes very different opinions. One participant clearly described App Inventor as an ‘unnecessary diversion’ from the challenge of learning about the key technologies that were introduced in the earlier parts of the module. Another participant stated that ‘[with] App Inventor there is a change of gear, you know, there is this flow chart and actually this is not so bad … it is then a lot easier’. (The flow chart refers to the graphical format of a program; ‘it’ refers to the task of understanding and working with programs.)

The inclusion of App Inventor exposes debates that are similar to those that surround the use of BlueJ in computing education (BlueJ, 2016): namely whether experienced and skills gained by using an educational environment can be easily transferred to industrial and commercial settings. Put another way, participants asked: ‘is the module team using the right tool?’ Another participant said that to ‘produce the apps, you need to have an understanding of how Android works’, which suggests that it might be useful for students to have a more detailed understanding of computer operating systems beyond what is introduced in the earlier module, *TM129 Technologies in practice*.

All tuition is online: students never get to physically meet their tutors. All tutorials are delivered through the university tutorial tool, OU Live. From the

interviews, it is clearly evident that there is a wide variety of ways in which participants use the tool. Some participants described using it to deliver a presentation that is based around a deck of slides prepared using PowerPoint. Other participants described adopting a more dynamic approach, actively choosing to share code with students, helping students to see what code looks like and how it might be manipulated using different tools. One participant reported that they 'always use the shared desktop and I have got two monitors so I pull my PowerPoint show in and out of the shared section and I will edit code live and I will go to websites live in the shared desktop'.

An interesting difficulty that many participants reported is the challenge of trying to get students talking. One participant said that 'the [OU Live] chat box can be much more popular than talking'. Another stated that 'many of them are quite reluctant to admit that they have microphones, but they will chat away [using the chat box] quite happily and that works quite well.' Aware of these challenges, another participant reported that they deliberately stay silent until students decide to make contributions. This implicitly echoes the work of Haythornwaite et al. (2000) who explored the importance of developing community in online and distance education.

Most participants record their OU Live sessions, but there are some who deliberately choose not to. This difference in practice is, to some extent, a reflection of the ambiguity in faculty policies. At the time of the study, the recording of online tutorials was not yet an official faculty recommendation, although some line managers of tutors did explicitly recommend recording, citing the view that recordings can help students.

Another important set of teaching tools are the online discussion forums. Students have access to a number of module-wide forums (also known as national forums) in addition to a forum for their tutor group. The module-wide forums are

moderated and supported by tutors who have been chosen by the module team, but all tutors can make a contribution if they choose to do so. Participants reported that, through the moderators, the module-wide forums offer 'a very efficient way of providing technical help' and that 'they are very supportive'. One participant said that it is sometimes difficult to follow the many discussions that take place in the module-wide forums: 'so I always say to [my students] if you want me to see your posting in the national forum send me an email and tell me so that I can go and have a look at it.'

Tutor group forums (TGFs) are used differently to the module-wide forums. One participant reported that discussions are difficult to maintain due to a lack of a 'critical mass' of students. Another said that their 'students tend to come to the national forum rather than the TGF'. An important comment from one participant was that the tutors 'use it as a noticeboard; I use it for sharing the results of the online sessions'. This was echoed by a further participant who stated that they use their forum to follow up after OU Live tutorials, and to remind students when they are going to take place.

Moving away from tools and the specific technologies that are taught, participants offered comments about the structure of the module. One important point related to the amount of materials that students were required to study. In relation to this, one participant said that 'I would make it 60 credits' and another that the 'breadth [of study] is right … [the] depth isn't there'. This is reflected in a participant opinion that the module could spend more time teaching programming skills and concepts and 'explaining how things work'.

Since the tutors represent the face of the university, the support that they are given from the module team is very important. All participants reported that they were happy with the marking and correspondence tuition guidance they were offered. One criticism was that some sections of the tutor notes lacked model answers. That said,

participants felt the module team responded quickly to any issue that arose when the module was being presented.

Discussion

Given the focus of *Web technologies* it is perhaps unsurprising that some participants reported that students can find the technical sections particularly challenging. This point can be related to the wider question of: how do we best teach computer programming? This is especially difficult to answer since we are teaching through distance education. This forces us to ask another question: what pedagogies are most appropriate? This question is especially important since students are not able to sit next to each other in a computer laboratory where they could provide each other with support and inspiration, ask each other for advice and attract their tutor's attention to develop and correct their understanding of how a technology or programming language might function.

The participant interviews have yielded some interesting insights. One view is both simple and compelling. One tutor stated: 'I think there should be more movies generally demonstrating things'. In fact, some have even taken it upon themselves to create their own video resources to fill an important need. Videos or screencasts can be used to illustrate how different tools can be harnessed and to familiarise students with the syntax and form of source code. Ahmadzadeh et al. (2007) suggest that exercises to teach debugging have the potential to help students learn programming. Videos could be created to illustrate debugging approaches to help students become familiar with different troubleshooting strategies and the context in which they can be applied.

As mentioned earlier, it was apparent that some participants chose to show students how to work with source code through OU Live sessions. In some respects, these demonstrations represent 'programming as performance', which is sometimes known as 'live coding' – situations where performers manipulate software code to

either demonstrate skill or ideas (Collins et al., 2003). Paxton (2002) suggested that live coding could become an important and useful pedagogic approach for the teaching of programming.

Addressing another aspect of digital pedagogy, the fact that many students remain steadfastly silent during OU Live sessions raises interesting questions, such as what might be done during earlier levels of study to reduce reticence to speak and what tutors could do during tutorials sessions to increase participation. These research questions go beyond the current focus of the project reported in this paper since they relate to the importance of engagement and community; but these are themes that are important to digital tutors and educators from all disciplines.

It was also interesting to note that while the case studies used in the module did feature in the interview transcripts, they were not discussed in any great length. This finding is surprising given the central role that they take within the module. An explanation for this might be that the participants were satisfied with their design, and preferred to highlight the more significant challenges that are faced by students.

The research reported here represents only one perspective of the work of a single cohort of very experienced tutors. The results and this accompanying discussion are, of course, substantially influenced by the experience of the researchers: the two tutor interviewers and the lead author. This experience can affect not just how the original interview data were coded, but also how the key themes that were identified have been read and interpreted.

One of the key research questions of whether the earlier modules help to prepare students for study at a higher level remains unanswered. Rather than speaking with tutors it may be more useful to speak directly with the students. What has been exposed during this project, however, is a broad range of valuable practices and experiences.

Conclusions

The paper began by presenting three interrelated research objectives:

(1) Are *Web technologies* students sufficiently prepared?

(2) Which aspects of the module do students find difficult?

(3) What could be done to improve or enhance the module?

By undertaking this research and by exposing the tutor's voice, it was also envisaged that more could be learnt about the challenges faced by both the students and tutors.

The first research question remains unanswered. Nevertheless, the tutor voice was clearly exposed and participants clearly expressed the view that students found some technical aspects challenging. They also suggested that students were uncertain about the operation of different pieces of technology and were challenged by the use of different sources of information. Participants demonstrated a wide variety of pedagogic practice when using online tutorial tools such as OU Live. They also presented different opinions about how online discussion forums were used, and expressed views about the structure of the module and what kind of materials might be useful for students.

This research has focused on listening to the tutor voice. From the beginning of this research, tutors were very willing to come forward and share their opinions. It is the author's view that listening to the tutor's voice is very important, if not essential, if we are to develop effective computing, IT and STEM distance education. Another voice that has not been studied in this study is that of the student's. The student's voice, along with the tutor's, needs to be listened to if we are to gain a full understanding of the challenges that accompany teaching technical subjects such as *Web technologies*.

References

Ahmadzadeh, M., Elliman, D. and Higgins, C. (2007) The impact of improving debugging skill on programming ability, *Innovation in Teaching and Learning in Information and Computer Sciences*, 6:4, 72–87.

BlueJ (2016) *BlueJ*, http://www.bluej.org/ Accessed 17 March 2016.

Collins, N., McLean, A., Rohrhuber, J., and Ward, A. (2003) Live coding in laptop performance, *Organised Sound*, 8:3, 321–30.

Hammersley, M. and Atkinson, P. (1995) *Ethnography: Principles in Practice*, Psychology Press.

Haythornthwaite, C., Kazmer, M., Robins, J. and Shoemaker, S. (2000) Community development among distance learners: temporal and technological dimensions, *Journal of Computer Mediated Communication*, 6:1.

Marshall, C., and Rossman, G. B. (1999) *Designing Qualitative Research*, Sage publications.

MIT (2016) *Scratch*, https://scratch.mit.edu/ Accessed 15 March 2016.

Nandi, D., Hamilton, M. and Harland, J. (2012) Evaluating the quality of interaction in asynchronous discussion forums in fully online courses, *Distance Education*, 33:1, 5–30.

parkrun UK (2018) parkrun, http://www.parkrun.org.uk/ Accessed 4 May 2018.

Paxton, J. (2002) Live programming as a lecture technique, *Journal of Computing Sciences in Colleges*, 18:2, 51–6.

PHP (2018) *PHP*, http://www.php.net/ Accessed 4 May 2018.

QSR (2016) *NVivo*, http://www.qsrinternational.com/ Accessed 17 March 2016.

Robins, A, Rountree, J and Rountree, N. (2003) Learning and teaching programming: a review and discussion, *Computer Science Education*, 13:2, 137–72.

The Open University (2018) *TT284 Web technologies*, http://www.open.ac.uk/courses/modules/tt284 Accessed 4 May 2018.

Appendix 1: Tutor interview plan

This appendix contains the introductory script that is read to participating tutors, along with a copy of all the questions that are asked during the participant interviews.

Thank you for volunteering to participate in the project. The aim of the project is to learn about what works well and what doesn't work well in terms of TT284 and the teaching of programming. Other objectives are to learn about what the university might be able to do to help you with your tutoring work, and to learn lessons that could feed directly into the design of future modules.

This interview should take up to an hour, and you will not be paid. The interview is also going to be recorded. Are you okay with this?

The interview will take the form of a series of open ended questions that are designed to ask you about your opinions and experience. There are no right or wrong answers to these questions. You are free to stop or pull out of the interview at any time. If you are unsure about any of the questions, please feel free to ask your own questions to clarify what is being asked.

Following the interviews, all recordings will be transcribed, and these transcriptions will be anonymised and then analysed. There is a possibility that some fragments of the interviews might be quoted in either internal reports or publications. No names will be attributed to any quotes.

If, by the end of the interview, you would not like your interview to be used as a part of the research project, this is okay too. You can either mention this during the interview, or send us an email afterwards.

Are you okay for the interview to start?

Questions

First of all, could you say something about yourself and your background? For example, what is your experience of being an Open University tutor, and what other experience of working with or teaching web technologies do you have?

What are the biggest challenges that you face as a tutor on TT284?

What do you think are the biggest challenges that your students face?

Can you give any examples of why you think this is the case?

What aspects of web technology do you consider that students find the most difficult?

What is your opinion or experience of teaching programming in TT284?

Are there certain subjects or skills that you feel that students who study TT284 may be lacking? Or, put another way, are there any areas that TT284 should perhaps teach or focus on that it doesn't at the moment?

Students gain access to learning through different parts of the module website and different devices. Are there aspects that work better than others? Are there other opportunities that you think could be taken advantage of?

TT284 obviously doesn't have any face-to-face sessions. Can you tell us about your views of the challenges and opportunities that OU Live offers students and tutors?

Could you say what works well and what doesn't work well in TT284?

Acknowledgements: I would like to thank the hard work of the two co-researchers: Dave Macintyre and John Williams. I would also like to acknowledge the authors of Web technologies and those who support its delivery. These include Soraya Kouadri, current module chair; John Busvine, TT284 curriculum manager; Neil Simpkins, production module chair; and module authors Doug Briggs and Nick Heap. I would also like to thank the Open University eSTEeM project, led by Clem Herman and Diane Butler, which funded this research. I also thank Emma Elder and the two reviewers for their invaluable suggestions; their comments and their time has significantly helped to improve this article.