The Role of Gamification in a Software Development Lifecycle

Neil Gordon[1], Mike Brayshaw[2], John Dixon[3], Simon Grey[4], David Parker[5]

[1]Department of Computer Science & Technology, University of Hull,
Cottingham Road, Hull, HU6 7RX, U.K.
[1]n.a.gordon@hull.ac.uk
[2]m.brayshaw@hull.ac.uk
[3]john.dixon@hull.ac.uk
[4]s.grey@hull.ac.uk
[5]d.j.parker@hull.ac.uk

Abstract

Teaching Software Engineering students raises a number of challenges; in particular that student developers typically demonstrate behaviours that run counter to good software development. These include failing to plan properly, failing to develop their software in a structured manner, and failing to meet specified deadlines (so called "student syndrome"). Consequentially, students exhibiting these behaviours are more likely to disengage from their studies. Even where submissions are made, they tend to be lower in quality, and may not demonstrate the true capabilities of the individual. Such alienation and disengagement is amplified by the current context of learning in a pandemic, with a wall of digital communication technology coming between teachers and learners.

In this paper, the authors will identify how gamification approaches can be applied to software development education, and how they can help to better motivate and educate future software developers through computer managed delivery and assessment. As motivation is a key factor, motivational properties known in computer gaming are applied within the new context of a software engineering lifecycle. The role of intrinsic and extrinsic motivation for developers is considere.

The gamified techniques identified are further enhanced with an Agile type approach. This has been particularly critical during 2020/21 where the shift to fully online learning for previously face to face taught students has placed new pressures on students and staff. A

feedback-led rapid prototyping style of teaching that allows for adaptive and effective teaching practices is also described. Finally, complimentary case studies on the use of approaches within a university environment are evaluated.

Keywords: Gamification, student syndrome, computing education.

1.0 Introduction

This paper considers an approach to the education of university Software Engineering students that attempts to blend Gamification, Agile Software Development Practices, and Flexible Learning with feedback-led rapid prototyping. This is in response to the behaviours of novice software developers, such as student syndrome [1], and how we can look at the role of intrinsic and extrinsic motivators [2]. We also consider some of the environmental factors of the 2020/21 Covid-19 pandemic and the benefits of an Agile approach [3].

In order to set the context, we shall start by outlining each of the component approaches and methodologies. We then show how we have used them, both individually and in blended form, to a range of Undergraduate and Masters courses within the current context using an action research methodology. Finally, reflection and evaluation is presented that shows from an action perspective what we have learned as researchers using these interventions and how we have changed the outcomes and experiences in each of the case studies for our students.

1.1 Engagement

Given the context of this research one of the key issues is getting students involved and engaged in their learning. The rapid move to online teaching presents a massive set of problems, both practical in terms of hardware and connectivity but also engagement with the material to be taught. The move from Face2Face to online means that we must look at new means to engage our students. These techniques must also scale for large cohorts of students (our largest cohort here was 150 students). The approaches that we adopt in this paper are, in part, a direct response to that challenge.

Engagement with learning has a wide variety of descriptions and definitions [4], though generally is linked to how motivated students are, as evidenced by the effort and time they apply to their studies. Finding ways to engage students is thus linked to motivating students to actively work on their own learning. One significant challenge of engagement is in identifying activities and evidence by which it can be measured, but which do not have undesired effects themselves An example of an undesired effect is where students respond to measures such as attendance monitoring or monitoring of opening a resource, when the students know they are being observed and may behave in an affected manner e.g. to simply register then leave an event, or to open then immediately close a resource [5].

1.2 Gamification

One of the challenges in Higher Education – arguably more so than in earlier stages of education – is that engagement can be considered optional by the student. This can result in a lack of commitment to learning opportunities and a reluctance to complete formative or summative work. Conversely, one of the desirable attributes of games is that successful games are engaging by definition. The only purpose of a game is to engage the player in play. Deterding proposes a definition of gamification as *"the use of game design elements in non-game contexts"* [6]. Gamification in education looks to apply some of the effective cues and mechanisms from games to engage students. Approaches that can be effective include allowing for multiple attempts, giving immediate feedback, and providing some form of reward (such as points, badges and leader boards). For a more complete analysis of how games mechanics map onto Higher Education teaching practices, see [7].

At its core, gamification provides a layer of extrinsic motivation to engage in activities that move towards a desired outcome. McGonigal identifies some common traits in games, namely having a goal, rules, a feedback system, and voluntary participation [8]. In the educational context this can map on to learning outcomes and how to do this is achieved, feedback could be via marks or game rewards (e.g. Badges, Levels, Awards, and Prizes) and enthusiastic participation in the learning activity. Issues can arise when the extrinsic motivation is not well aligned with the underlying goals of the activity. For example, Visaria et al [9] describe how a student's active engagement with an attendance monitoring system does not necessarily lead to their active engagement with teaching material. It could be argued that summative assessment of a student's knowledge or ability is separate from a student's process of learning. In that case, the assessment provides an additional extrinsic goal and a feedback system through and summative marks.

Perhaps more worryingly, Pink [10] warns of a number of undesirable effects of extrinsic motivators – including a reduction in intrinsic motivation. That is to say that if a student is intrinsically motivated to learn about a particular topic, then giving them an extrinsic motivator, like assessment marks, may reduce their existing intrinsic motivation. By offering students marks for engaging in learning they may become less likely to learn independently for the sake of learning alone than with additional incentive.

Psychologist Csikszentmihalyi recognised the concept of flow as a state of optimal engagement [11], identifying that in order to achieve flow through a given activity, the challenge presented by that activity must be well suited to the ability of the individual engaged in that challenge, and that individual must receive immediate feedback through their engagement with the challenge – improving their skill through deliberate practice. This indicates a need for flexible and personalised learning.

1.3 Agile Development

Our next approach uses Agile Software Development. As a software engineering methodology, agile can be characterized as an iterative development with self-organising teams that explore the problem and build a solution (see Figure 1). This allows for the incremental delivery of a product [12].

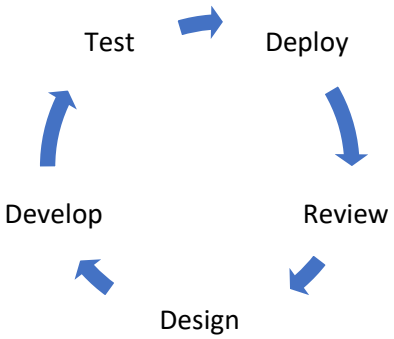

Figure 1: Agile Development

Agile is distinct from some traditional software development methodologies, which generally follow a more linear process with a full problem statement being used to define the requirements, that then lead to the product being developed, which is then tested (see Figure 2).

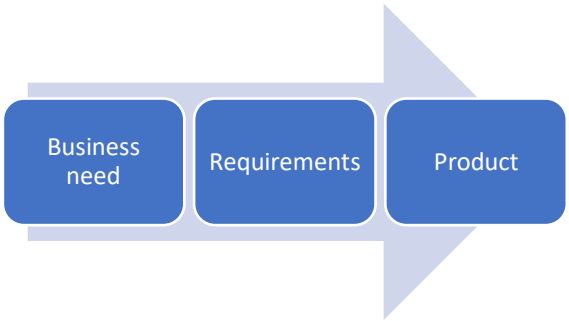

Figure 2: Traditional Software Development

Agile development can also be framed as a formal process to gamify software development. A backlog of desirable software features represents a daunting software development challenge. This challenge is broken down into more manageable smaller parts known as sprints, thus ensuring that each part is at an appropriate level for the software development team to be productive. Feedback is built-in to the process on multiple levels of resolution. The adoption of test-driven

development practices gives individual developers confidence that they are implementing features without inadvertently introducing problems later on, burn down charts give feedback on the productivity of the team over each sprint, and allows the level of challenge to be adjusted to suit the skills of the team. Overall development progress is clear as the backlog of features to be implemented is reduced, and the list of features that have been implemented increases throughout the project.

1.4 Flexible Pedagogy

Flexible pedagogy provides a framework for considering how to make learning truly student centric [13] with the intent to give students choices in how, where, and when to study. From a practical perspective, enabling these choices requires suitable tools, and can be provided with the appropriate use of technology, enabling students to access resources and be assessed in flexible ways [14].

1.5 The Student Context and Delays in Work

Two key problems for software development are student syndrome, and Parkinson's Law. As noted earlier student syndrome is characterized as leaving tasks until close to a deadline, waiting until the last moment when an action is required [15]. This is also a known phenomenon in software development, and can be problematic within agile processes [16]. In an educational setting, getting it wrong can lead to missing deadlines and potentially failing the task, a module and even degree programme [17]. Parkinson's Law [18] notes that work often expands to fill the available time and resource. The impact of this on student software development is considered later.

1.7 Covid-19

One of the much-used terms by governments and institutions – including in higher education – beginning in 2020 has been that of being agile in response to the COVID-19 crisis, though not necessarily in an appropriate nor successful way [19]. The 2020/21 pandemic has seen the shut-down of face-to-face teaching across institutions and across the world, with a rapid and unexpected cessation of standard teaching. One approach used in education to adapt to and respond to these challenges has been to adopt a flexible learning approach, facilitated through technology [20]. This has raised issues about access to computer technology and has raised the requirement for minimum level of internet access. Institutions have adopted online only learning, often adapting assessment as well as other temporary fixes, including passing/condoning students without necessarily any validation that students have completed work.

2.0 Teaching and Software Development

Keeping up to date with the rapid pace of change in the technology sectors sets a particular challenge for Computer Science higher education providers. Student expectations generally tend towards the cutting-edge to boost degree usefulness [21], with obsolete or outdated topics causing significant student dissatisfaction [22].

Employers also actively seek graduates with up-to-the minute skills, requiring quick adaptation of university courses [23]. We will explore how Agile is useful for the educator; however, it also provides a visible exemplar for students who are likely to need to utilise Agile techniques in their professional practice.

2.1 Agile Module Development

With time at a premium for many educators, redeveloping a full course or module to take advantage of the advances in the domain is prohibitive. Even if this action is taken, large swathes of the content is likely to be outdated within a short timeframe. By applying techniques from the Agile toolkit, the module can be constantly redeveloped on a rolling cycle. Regular evaluation and feedback from students provide a dynamic "to-do list" (backlog) from which items can be taken as time allows. A large group of students providing feedback can provide a valuable snapshot of importance, allowing item prioritisation and ensuring that focus is given primarily to areas where learners are most keen for development. As a side-product, areas which regularly slip to the bottom of the prioritised list may indicate concepts which are 'timeless' (e.g. general theory) and therefore not in need of update. Alternatively, and especially where feedback indicates disinterest, low-priority items can be evaluated for removal where they may have become redundant.

2.2 Engagement via Collecting Feedback

Several techniques are applied to collect feedback and ideas from students as a means to enhance engagement. End-of-semester anonymous reviews provide an opportunity for Likert questions on quality and appropriateness of specific areas (assessment, course speed, etc.) whilst also requesting freeform feedback. This technique is valuable to help determine key areas to change in the subsequent year but is not rapidly responsive and has little impact on those who have provided the feedback. These students lose out on any benefit from providing their views and may therefore lack awareness of their participation in the Agile cycle. This subsequently reduces the likelihood of their engagement and increases dissatisfaction: "my feedback isn't acted upon". Partnering end-of-semester reviews with mid-semester reviews helps to increase both the quantity of feedback and the opportunities to add to and refine the backlog. In addition, by taking feedback before the content has all been delivered, high-priority changes can be made which positively affect the students' learning. To this end we applied Agile approaches to give flexible responses and rapid feedback to our students.

2.3 Closing the Feedback Loop

Closing the loop conventionally means to act upon the feedback, thus incrementally improving the value of the teaching, to inform students of actions taken or intended and to evaluate the effectiveness of actions taken [24]. By establishing this visibility

of positive and rapidly reactive change we are able to affirm the value in student involvement and show that their input has real impact on their teaching and learning. This, in turn, makes participation in providing future feedback more likely and therefore more strongly engages the students in the Agile process. Furthermore, given the value placed on student voice in the UK National Student Survey (NSS), awareness of how staff value and respond to feedback is also valuable from an institutional perspective. Proactive identification of where feedback has been acted upon has driven an increase in feedback quantity and quality and, where action is taken little and often, regular and perceptible change can be observed, announced and evaluated readily: "I'm returning to this topic based on feedback...", "This topic area was added as it was requested in feedback", "I have redeveloped last week's slides to add some more examples after some feedback, is this useful?", etc...

3.0 Modified Module Delivery

This approach was adopted by the authors in 2020 to tackle the problems faced with the rapid and unanticipated move to entirely online teaching for multiple modules, within a Software engineering programme. These include the following 5 action interventions (the complimentary blended approaches used are indicated in square brackets):

- level 4 (first year) approximately 150 students on mathematical underpinning, and also on professional and legal requirements [Gamification, Flexible Learning]
- level 6 (3rd year/honour's stage) approximately 40 students on advanced software engineering [Gamification, Flexible Learning]
- level 6 (3ed year/honour's stage) 80 students on Distributed Systems [Agile style interaction and feedback]
- level 7 (Masters) approximately 20 students on component-based software engineering [Agile Methodology and feedback]
- level 7 (Masters) approximately 30 students on Commercial Development Practice [Agile Methodology and feedback, Gamification]

The teaching is based on an agile cycle and the following highlights the key aspects mapped in to the categories already explored.

3.1 Engagement

As noted earlier, engagement is a challenge in higher education, and can be especially problematic in large cohorts. One way to address this is to use data and indicators to help identify where students are failing to engage and may need prompting. In the context of online delivery, this can be through making use of learning analytics, with a focus on where students are actually doing activities. For example: attempts at formative tasks and quizzes, how far they are contributing to community activities such as discussions, and whether they appear to be progressing.

For the modules this year, one approach that worked well was to include formative quizzes so students could indicate their progress with programming and other activities within their modules.

3.2 Gamification

Some gamification approaches to engagement include using quizzes that align with the taught material, and which give immediate marks back to students to help them understand their own progress, akin to the score/high score concept within a game. Utilising banks of questions and/or computer-generated questions, means that these can be done multiple times – similar to the approach of multiple lives/attempts in a game. High score tables (even where anonymous) can assist students in understanding what is possible – as their own ideas of this can be far removed and their own self-assessment of their performance can be inaccurate. Creating dependencies within module materials – so that students should complete one activity before being able to start the next, is also effective in assisting students in understanding the recommended learning journey: akin to many games where completion of some challenges opens up others.

3.3 Front-Loaded Teaching in 'The Tutorial Room'

In project modules, often the focus is on the development of a product or set of artefacts that are expected to be delivered over the course of many months. Much of the knowledge required to succeed in the project needs to be learned and practiced before it can be effectively applied. Rushing to start before key elements, such as project management and the impact of architectural decisions are fully understood, can lead to diminishing returns. Mistakes made early on are hard to rectify, particularly as deadlines loom and restarting becomes impractical. It may also be that the appearance of sunk-cost inhibits such changes. Front-loaded teaching and opportunities for practicing techniques in a consequence-free but intensive first few weeks ensures students have core skills ready to select and utilise when they are required. This has been applied successfully with approximately 30 Masters (level 7) students on a commercial development practice group project. The method is comparable with the concept of a tutorial room in digital gaming [25]. Players are exposed to core game mechanics in a safe environment and given opportunities to practice. This has two beneficial effects: the participant is made aware of what is possible and is then able to be more completely immersed in future tasks. Even where concepts are not fully embedded, to the point of immediate recall and application, exposure ensures awareness that facilitates a return to the taught content when required. It is therefore critical that teaching material is highly available and easily searchable. This is achieved through upload to online digital learning technologies and appropriate signposting in lectures, workshops and forums. Completing the front-loaded section of the module quickly ensures that students can start to make concrete progress on the project. However, two conflicting issues arise, especially when taught content is online: boredom or disengagement leading to slow progress through the tutorial room and a delayed project start, or rushed tutorial room with little retained understanding. Gamification techniques were applied, including:

- providing rewards for progressing through the content (score/high score and rewards – e.g. valuable project information released only upon completion)
- introducing dependencies and barriers to continuation (minimum quiz score before moving on to next section)

This helped to maximise engagement and prevented rushing and superficial learning. Additionally, progress through content is easily monitored by checking quiz returns and number of attempts. This facilitates identification of struggling learners who may benefit from intervention.

3.4 Agile Development

The agile approach was helpful in managing modules and programmes within the context of Covid-19, where teaching and assessment had to be adapted, thus requiring a combination of review, planning, developing and then delivering. This often came in to tension with institutional quality approaches as considered below.

In some modules, especially those with large scope and where technologies move quickly (e.g. level 6, 3rd year/honour's stage, approximately 80 students on distributed systems), these two mechanisms have been partnered with a permanently open, simple and (optionally) anonymous feedback collection system. This online survey allows learners to give immediate observations and comments – both positive and constructive. The existence of this option is reintroduced at the start of every taught session and students are invited to participate proactively in the rapid Agile redevelopment of the teaching.

3.5 Delays in work

When developing software, students frequently exhibit behaviours that are contrary to effective project management, including where this intersects with carrying out other aspects of their studies. This demonstration of "student syndrome" is shown where they hand in work at the last minute, or even beyond the deadline which can lead to penalties (including zero marks being awarded). For some, this problem is exacerbated by the lack of the formality and routine of face-to-face campus learning. Providing recommended due dates to supplement the actual final deadline can help, though there are still many students who are missing the deadlines.

3.6 Flexible Pedagogy

Applying flexible pedagogy in the online variants of the modules subsequent to the Coronavirus lockdown, there are several aspects that were implemented that provide flexibility:

- Where: material and class interaction being provided as webcasts, allows students to participate where they wish;

- When: the recording of sessions allows for flexibility in when to participate, along with the ability to pause and replay, allowing for control of pace. This flexibility of pace was also applied to assessment, with a series of fixed automatic extensions: so students who missed one deadline would automatically be allowed to submit at the next one;
- How: offering a variety of resources, from live webinars, recorded content, written materials, and interactive workbooks all offer a variety of approaches for students to choose what works most effectively for them. Again, this also applied in some cases to assessment: as time progressed and opportunities for assessments varied, students had different assessments made available. For example, team projects in some modules became individual projects (with elements to demonstrate how to organize and plan for team activity, even where that does not happen in practice).

3.7 Quality Processes for Education

The agile approach to teaching is in direct tension with the more linear approach that institutions tend to use for their programme and module update and approval processes. These processes reflect the long-standing nature of a degree: but make rapid responses difficult, and can be a barrier to innovation and change. Another challenge with traditional educational quality processes is that it can constrain assessment practices: especially when it comes to offering alternative assessment choices by a student.

3.8 Covid-19.

The Coronavirus pandemic of 2020/21 has highlighted the challenges, where institutional responses to the rapidly evolving situation tended to generate more bureaucracy, with forms to plan mitigation on delivery and assessment that fell behind the rate of change of other activity. Moreover, planning ahead meant that decisions on timetabling, delivery and assessment patterns were being made prior to any review of the current experience. This emphasizes the need to be able to adopt a truly agile approach to module and programme delivery to allow rapidly adaptive responses. The challenge is how to do that without reducing quality.

4.0 Evaluation

The following table illustrates how we have evaluated the changes that we have made using our blended approaches. For each cohort we have indicated the approaches taken for that particular cohort. We have then indicated the outcomes of these approaches both in terms of the researcher(s) carrying out the changes but also for the outcomes of the students who were participants in this change.

Table 1: Gamification and Flexible Interventions.

Student Cohort / topic	Intervention Action	What the Researcher Learnt	Outcomes for Students
Level 4 (first year). Approx. 150 students on mathematical theory, as well as professional and legal requirements	Gamification, Flexible Learning	Student engagement improved with the used of quizzes with rapid feedback. Helpful for staff to monitor engagement, and to contact those who seemed to be dis-engaged.	Students found the feedback helpful, and improved regular work. The opportunities for different learning resources (flexible) supported a wider range of students.
level 6 (3ed year/honour's stage) 80 students on Distributed Systems	Gamification, Agile Interaction and Feedback	Student engagement increases with clear evidence that the tutor is tracking progress and achievement through weekly quizzes. Students who do poorly or do not participate are easy to identify and support. Feedback quantity and quality improves when students are more integrated into the process.	Satisfaction improves when iterative, perceptible module improvements are based on student feedback and priorities: a year-to-year 24% increase in average overall satisfaction and 47% increase in the "Staff have made the subject interesting" category were recorded when this approach was applied.
level 7 (Masters) approximately 30 students on Commercial Development Practice	Gamification, Agile Interaction and Feedback	Students' progress through a 'tutorial room' section of front-loaded teaching, are able to make faster, more error-free progress on projects. Engagement and fast progress is particularly noticed when barriers to progression (e.g. dependencies and quizzes) are partnered with rewards for progression.	Agile is a core component of a degree associated with software development. When asked to rate their understanding of Agile at the start and end of the module, Students registered a 41.7% improvement. Formal teaching is restricted to one intensive front-loaded week but 97.5% of students agreed that they have improved due to their experience in this module.

4.0 Conclusions

Our aims were to raise the bar both in terms of experience of students and outcomes for them in learning Software Engineering. We did this with Gamification taking a core role but being used in a blend with other techniques like Agile Development, Flexible Learning, and the considered use of Feedback. From the outcomes noted in the above table we can note that we are able to demonstrate that engagement can be improved, with corresponding improvements in retention and attainment.

In regard to the agile model for module (and program) delivery we found that it can be an effective way to manage education during a crisis: though that needs support from institutional processes and systems to allow for academic judgement and for truly adaptive and agile approaches.

What the whole study has done is show that Gamification, Agile Software Development Practices, and Flexible Learning with feedback-led rapid prototyping as techniques in their own right can make valuable contributions to the pedagogical mix. They were also shown to be valuable in the rapid changes caused by the pandemic. What will emerge after COVID is an interesting question? Many of the changes, for example in flexible learning, that have been necessitated by the global situation have situation have worked well. So, the resultant modus operandi that emerges may be a mix of both a return to traditional forms of teaching but also the retention of techniques that proved themselves in the crisis. We have here argued how combining flexibility, gamification, and agile interaction can have a role in this mix.

5.0 References

1 Smith, D. (2010) The effects of student syndrome, stress, and slack on information systems development projects. Issues in Informing Science and Information Technology, 7, pp.489-494.

2 Rigby, S., and Ryan, R.M. (2011) Glued to Games, Peaeer (Oxford, England), ISBN 978-0-31336225-5

3 Laanti, M., Similä, J., & Abrahamsson, P. (2013). Definitions of agile software development and agility. In European Conference on Software Process Improvement (pp. 247-258). Springer, Berlin, Heidelberg.

4 Groccia, J. E. (2018). What is student engagement?. New Directions for Teaching and Learning, 2018(154), 11-20.

5 Gordon, N.A. and Grey, S. (2018) Approaches to Measuring Attendance and Engagement. New Directions in the Teaching of Physical Sciences, (13).

6 Deterding, S., Dixon, D., Khaled, R. and Nacke, L., 2011, September. From game design elements to gamefulness: defining" gamification". In Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments (pp. 9-15).

7 Gordon, N., Brayshaw, M. and Grey, S. (2013) Maximising gain for minimal pain: Utilising natural game mechanics. Innovation in Teaching and Learning in Information and Computer Sciences, 12(1), pp.27-38.
8 McGonigal, J., 2011. Reality is broken: Why games make us better and how they can change the world. Penguin.
9 Visaria, S., Dehejia, R., Chao, M.M. and Mukhopadhyay, A., 2016. Unintended consequences of rewards for student attendance: Results from a field experiment in Indian classrooms. Economics of Education Review, 54, pp.173-184.
10 Pink, D.H., 2011. Drive: The surprising truth about what motivates us. Penguin.
11 Csikszentmihalyi, M. and Csikzentmihaly, M., 1990. Flow: The psychology of optimal experience (Vol. 1990). New York: Harper & Row.
12 Laanti, M., Similä, J., & Abrahamsson, P. (2013, June). Definitions of agile software development and agility. In European Conference on Software Process Improvement (pp. 247-258). Springer, Berlin, Heidelberg.
13 Ryan, A. and Tilbury, D. (2013) Flexible pedagogies: New pedagogical ideas. Higher Education Academy, London.
14 Gordon, N. (2014) Flexible pedagogies: Technology-enhanced learning. The Higher Education Academy, pp.1-24.
15 Shoen, F. (1998) Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies. Journal of the Operational Research Society, 49(2), pp.183-184.
16 Smith, D. (2010) The effects of student syndrome, stress, and slack on information systems development projects. Issues in Informing Science and Information Technology, 7, pp.489-494.
17 Gordon, N.A. (2016) Issues in retention and attainment in Computer Science. York: Higher Education Academy.
18 Gutierrez, G.J. and Kouvelis, P. (1991) Parkinson's law and its implications for project management. Management Science, 37(8), pp.990-1001.
19 Janssen, M. and van der Voort, H. (2020) Agile and adaptive governance in crisis response: Lessons from the COVID-19 pandemic. International Journal of Information Management, p.102180.
20 Huang, R.H., Liu, D.J., Tlili, A., Yang, J.F. and Wang, H.H. (2020) Handbook on facilitating flexible learning during educational disruption: The Chinese experience in maintaining undisrupted learning in COVID-19 Outbreak. Beijing: Smart Learning Institute of Beijing Normal University.
21 Giannakos, M. N., Pappas, I. O., Jaccheri, L., & Sampson, D. G. (2017). Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. Education and Information Technologies, 22(5), 2365-2382.

22 Kasurinen, J. (2018). Software Engineering Expectations. Proceedings of the 2018 Workshop on PhD Software Engineering Education (SWEPHD2018)
23 Chetwynd, F., Aiken, F., & Jefferis, H. (2018). Reflections on the 2017 HEA STEM conference: graduate employability challenges and solutions. Higher Education Pedagogies, 3(1), 490-494.
24 Shah, M., Cheng, M., & Fitzgerald, R. (2017). Closing the loop on student feedback: the case of Australian and Scottish universities. Higher Education, 74(1), 115-129.
25 Chiapello, L. (2016). From “spectator knowledge” to “pragmatic knowledge”: how a philosophical understanding of knowledge can help create better video game tutorials. Communication présentée à la conférence The Philosophy of Computer Games Conference: Knowledge, Valletta, Malta.

Biographies

Dr Neil Gordon is a senior lecturer in the Department of Computer Science & Technology at the University of Hull, where he is the programme leader for the Computer Science family of degrees. His research area includes the interface of mathematics to computer science, as well as computing education. He has produced a number of national reports related to Higher Education, including issues of retention and attainment in computer science.

Dr David Parker is a lecturer in the Department of Computer Science & Technology at the University of Hull. He has taught at many levels of higher education from introductory software development to master’s level. He takes particular interest in applying innovative approaches to improving student engagement in learning.

Dr John Dixon is a computer science researcher, software developer, software engineer and lecturer. His current work at the University of Hull is conducted across undergraduate, Masters and PhD levels, and encompasses lecturing, support, mentoring and scholarship. He also works closely with academic, business and industry professionals on commercial projects. John specialises in the areas of networking, connected services and distributed systems, with particular expertise in C# and .NET.

Simon Grey is a lecturer in the Department of Computer Science & Technology at the University of Hull, where he is the programme leader for the Games Development family of degrees, and teaches games programming at all levels, from pre-certificate to post-graduate. He has an interest in delivering effective and scalable teaching, and motivating engagement in teaching through gamification.

Dr Mike Brayshaw was a lecturer in the Department of Computer Science & Technology at the University of Hull. His main interests are in Artificial Intelligence, HCI, and Computers and Education. To the latter end and in the context here he had worked on the psychology of programming, tools, execution models and environments for neophyte software engineers. He has 27 years of teaching programming at Undergraduate and Masters’ level