Goldsmiths Research Online

Goldsmiths Research Online (GRO)
is the institutional research repository for
Goldsmiths, University of London

Citation

Katan, Simon and Anstead, Edward. 2020. 'Work In Progress: Sleuth, a programming environment for testing gamification'. In: 2020 IEEE Global Engineering Education Conference (EDUCON). Porto, Portugal 27 - 30 April 2020. [Conference or Workshop Item]

Persistent URL

https://research.gold.ac.uk/id/eprint/28185/

Versions

The version presented here may differ from the published, performed or presented work. Please go to the persistent GRO record above for more information.

If you believe that any material held in the repository infringes copyright law, please contact the Repository Team at Goldsmiths, University of London via the following email address: gro@gold.ac.uk.

The item will be removed from the repository while any claim is being investigated. For more information, please contact the GRO team: gro@gold.ac.uk

Goldsmiths
UNIVERSITY OF LONDON

Work In Progress: *Sleuth, a programming environment for testing gamification*

line 1: 4th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 2nd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

***Abstract*— Sleuth is a gamified platform developed as a practical tool for teaching introductory programming to large student cohorts. It focuses on building fluency through repeated practice whilst developing syntactic and conceptual knowledge. The platform is currently used for online and campus-based teaching with around 1200 active students at any given time. We are currently using Sleuth as an iterative testbed for empirically testing theoretical considerations around gamification and automated feedback generation. Our ongoing research investigates effects of varying feedback types on student attainment and optimising student progression.**

Keywords—gamification, introductory programming, automated feedback, programming pedagogy

I. Introduction

Programming can be viewed as comprising a varied and interlaced set of high and low order skills which can be developed over time through regular practice [1]. However, facilitating practice presents the lecturer with numerous challenges; the production of sufficient unseen code exercises, meeting the volume of grading given the large size of modern undergraduate cohorts; detecting and preventing plagiarism, and motivating students to undertake repeated practice [2].

We have designed Sleuth as a practical teaching tool to address these issues. Sleuth is a series of gamified code puzzles themed around a film-noir detective story. Students play the character of a fledgling detective and are guided by 'the Chief' who gives them feedback on individual puzzle attempts as well as their general progress in the game. The platform uses automatic grading and exercise generation to provide instant feedback and an inexhaustible supply of puzzles for students to practice on.

Gamification is the use of game design elements in non-game contexts [3]. There is an emergent consensus around theory within the field, but despite numerous studies reporting positive impacts, there is a lack of empirical work validating theoretical conjecture [3], [4]. The context of deployment for Sleuth in which it is used simultaneously with multiple large cohorts, provides an opportunity for carrying out such research.

Given gamification's interdisciplinarity, our current research draws on theory comprising game design, computing pedagogy, assessment automation and theories of motivation. In particular we are investigating how student attainment is affected by varying the types of feedback provided by our automated system. This research direction is motivated by theoretical considerations from the literature as well informally collected data from logs of initial runs of Sleuth and module evaluations. Further topics include utilising game mechanics to influence learner behaviours, testing for optimal progression, and exploring the degree to which intrinsic motivation [5] is generated through the addition of ungraded, hidden, 'Easter egg' puzzles.

II. Context

Computer science pedagogy is widely recognised as problematic [1], [2]. It is the worst performing subject with regards to undergraduate non-continuation in the UK by a margin of more than 2%. 9.8% of the 2016/17 cohort of young undergraduate entrants, terminated their studies early [6].

In reviewing the pedagogical literature, Qian and Lehman classify difficulties pertaining to 'syntactic knowledge, conceptual knowledge, and strategic knowledge.' Syntactic and conceptual knowledge are presented as prerequisites for developing strategic knowledge. Deficiencies in the former prevent students from developing the latter. Students' 'lack of well-established strategies and patterns often leads to various challenges in planning, writing and debugging programs.' [1]

Additionally, in teaching first year students through lab-based activities we observe difficulties with mechanical skills such as accurately typing syntactical patterns, correctly selecting code blocks for cutting and pasting, making use of code completion in the editor, arranging screen space for effective editing and monitoring, managing files and folders, and effectively executing the save, compile and run workflow. A firm grounding in such skills are prerequisite to the development of syntactic and conceptual knowledge.

Thus programming can be viewed as comprising a varied and interlaced set of high and low order skills which can be developed over time through regular practice. However, facilitating practice presents the lecturer with numerous challenges; the production of sufficient unseen code exercises, meeting the volume of grading given the large size of modern undergraduate cohorts; detecting and preventing plagiarism, and motivating students to undertake repeated practice [2], [7].

Such issues are exacerbated by the modern university context in which staff increasingly teach large and heterogeneous cohorts of students with finite resources. In the authors' host institution, the Introduction to Programming cohort typically exceeds 300 students approximately 50% of whom have no prior programming experience with 10% reporting more than three years. Students to staff ratios are maintained at around 25:1 through a combination of academic staff and hourly paid teaching assistants. The emerging online context, where the authors run an online version of the module as part of a BSc Computer Science,

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

presents greater challenges still. Here cohorts exceed 600 students with a staff student ratio of 50:1 with one hour a week of student contact time.

III. State of the Art

Gamification is a nascent term becoming widely adopted in the second half of 2010. Whilst the various available literature reviews report inconsistencies in definitions and theoretical frameworks, Deterding et al's definition of 'the use of game design elements in non-game contexts.' has gained most traction [3], [4]. Research into gamification continues to grow and shows signs of institutionalising as a cross-disciplinary field [4], [8], [9]. Within education O'Donovan et. al highlight the existence of game-like attributes and advocate improving the 'university game' through game design techniques [10]. Indeed numerous applications of gamification within the educational context appear in the literature. Applications in the field of Computer Science are well-represented. Dicheva et al speculate that this is because "utilizing gamification assumes a certain type of environment that supports incorporating and visualizing the selected game mechanisms and dynamics" [9], [11].

In their seminal paper Deterding et al. classify 'game design elements' into five levels ordered from concrete to abstract; Game interface design patterns, game design patterns and mechanics, game design principles and heuristics, game models, game design methods [3]. Dicheva et al refine this model for specific use in the educational context arriving at a series of 'educational gamification design principles', not exclusive to games; goals/challenges, personalization, rapid feedback, visible status, unlocking content, freedom of choice, freedom to fail, storyline/new identities, onboarding, time restriction, and social engagement. Their mapping study identifies visual status, social engagement, freedom of choice, freedom to fail, and rapid feedback as most commonly employed within the educational context. Theories of intrinsic and extrinsic motivation, as grounded in self-determination theory developed by Ryan and Deci [5], are commonly cited in theoretical writings about Gamification, and this is reflective of the concern with effecting motivational and behavioural change within its practice.

The automation of feedback generation for assignments has been commonly applied in the teaching of computer programming, the practice dating back to the 1960s [12]. Such systems are helpful in addressing the aforementioned problems of facilitating sufficient programming practice in modern teaching contexts. Feedback from such tools can take a variety of forms including hints, visualisations and summative grades. Narciss sets out a conceptual framework for feedback in interactive instruction in which feedback is separated into components range from concrete types such as 'Knowledge of performance for a set of tasks' to higher-level categories such as 'Knowledge about Meta-cognition' [13]. Keuning et al specialise this framework for the domain of computer programming, for example subdividing 'Knowledge about mistakes' into 'Test failures', 'Compiler errors', 'Solution errors', 'Style Issues' and 'Performance Issues', and 'Knowledge about how to Proceed' into 'Bug-related hints for error correction', 'Task-processing steps', and 'Improvements.'[12] They identify that feedback from current systems is predominantly 'Knowledge about mistakes,' and that this feedback alone is not sufficient for all students to progress.

'Automated exercise generation' refers to the use of procedural content generation to produce on-demand variations of assignments [14]. The practice is less widespread than automatic grading. Sadigh et al take a template-based approach, identifying differentiating elements which can be parameterised and varied to create new exercises. Le and Pinkwart classify programming exercises into a number of types based on the degree of problem definition [15]. Class 1 exercises have a single correct solution, Class 2 exercises can be solved by implementation variants, whilst Class 3 exercises can be solved by applying alternative solution strategies.

In combining gamification, automatic feedback generation and automatic exercise generation, Sleuth falls into a niche sub-category with few examples of existing practice. Code Combat [16] is a comparable example which follows a themed hierarchical level-based design. Despite commercial-grade production values and 42 code puzzles within its first level alone, the puzzles themselves are static and produced without automation. In, Microsoft Research's Code Hunt [17] challenges are set by providing players with series of tests which match the functional behaviour of a secret goal algorithm to be deduced by the player. Although it was not designed as a teaching game but rather as a tool for practice and competition, Code Hunt nevertheless has a 'default zone' which takes students through a progressive set of puzzles covering arithmetic, loops, strings, arrays and so on. Despite the claim of 'impressive' figures, the retention rate of 85% per puzzle falls far below what would be acceptable at a University level. As with Code Combat, Code Hunt uses static puzzles. However, these are arranged in terms of difficulty by using player performance metrics.

More recent studies in Gamification have seen a maturation in terms of empirical study [8]. Whilst the first wave of studies attempted simply to establish the efficacy of Gamification, more recent studies have drawn on theoretical conjecture to investigate how particular design elements work. Sleuth's context, in which there are multiple simultaneous deployments running with large cohorts of students on a cyclical basis, provides us with a unique opportunity for theory-driven empirical studies carried out in the field. Through such studies we aim to investigate a range of topics whilst iteratively improving the design of Sleuth.

IV. Design

A. Design Description

Sleuth is a gamified assessment platform for learning programming using the p5.js framework which is designed around a media programming pedagogy (ie. teaching code through visually orientated application development). The platform is designed to facilitate opportunities for students to practice elementary programming techniques comprising, 2D drawing, variables, conditional logic, iteration, data structures, and functions.

In the game, students play a rookie detective working for the agency 'Sleuth & Co'. Using a web-app they gain points and unlock adventures by solving code crimes. A fictional entity called 'The Chief' provides students with feedback about individual attempts at solving code crimes as well as more general communication about their progress. Code

crimes are hierarchically organised into 16 cases, each based on a particular topic and consisting of four crimes to be solved. These are presented as a series of clickable badges arranged in an ordered grid. Students can attempt the cases in any order and need not complete them before starting another. However, the crimes of each case progress in order of difficulty and are unlocked in sequence as the student solves them. Once a crime has been solved a 'solved' tag appears on the top left-hand corner of the badge for that crime and proceeding crime is unlocked.

Students' grade comprises a rookie score and pro score. The rookie score is made up of the average of the first nine cases which are made available to students from the start of the course. At the midpoint of the course, the students 'go pro.' Their rookie score is frozen and the remaining seven cases are released. The pro score is made up of the average of all the cases.

Crimes take the form of template code to be downloaded and completed according to themed instructions. Students can upload their solutions multiple times for automatic grading and feedback, with the highest score being retained on each attempt. They receive immediate feedback from the Chief which includes any compile or runtime errors and tells them what parts of the task they have achieved and what parts they still need to work on. Students get five attempts to solve a crime after which the Chief suspends them from that particular case for one hour. Upon returning, students must begin afresh by downloading a new variant.

Such variants are produced through automatic exercise generation [14]. This approach provides the opportunity for repeated practice where necessary whilst reducing scope for plagiarism. Variation occurs by permuting selected parameters of a code template. Example parameters include task requirements, organisation of data, image dimensions and variable names. All crimes can be catagorised as 'Class 2' type problems which can be solved through implementation variants [15].

Fig 1 Shows two variations of case 101 - The case of Lina Lovelace. Students are tasked with writing code to draw a rectangle around a Lina Lovelace, the character wearing the red dress.

B. *Design Motivation*

The above specification was guided by a number of pedagogical concerns and game design considerations. Dicheva et al [11] define a framework of Educational Gamification Design principles which we have used to describe our design motivations in Fig 2 below.

There are nevertheless design principles from the framework which are out of scope the platform. Whilst

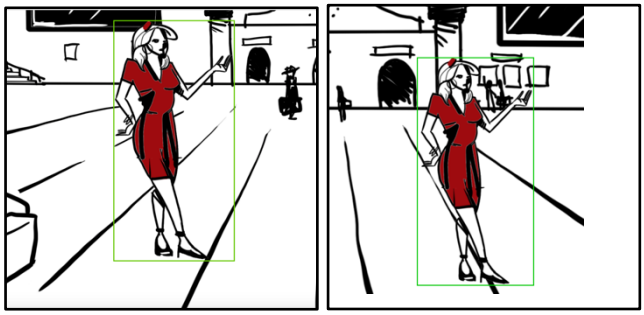

Fig. 1. 2 examples of Sleuth crime 101 showing varied puzzle generation

Design Principle	Features in Sleuth
Goals and Challenges	Solving code crimes and cases.
Rapid Feedback	Autograded feedback from the Chief.
Progress: visible progression to mastery	% Complete score for individual code crimes. Solved badges on crimes.
Accrual grading	Rookie and Pro scores.
Access/Unlocking Content	Hierarchical level design realised through Cases
Freedom of choice	Students are able to choose cases in any order
Freedom to fail: low risk from submission, multiple attempts	Use of exercise generation to provide variant puzzles.
Storytelling	Rich theming around 'Sleuth & Co' detective agency
New identities / roles	Students play the detective.

Fig. 2. Design motivations for the Sleuth platform

'competition and cooperation' suggests further research directions in terms of introducing leaderboards, and clans, others such as 'time restriction' are pedagogically inappropriate for this context.

V. Preliminary Findings

Sleuth was not designed as a research study, but rather as a practical teaching tool for improving the experience of campus-based and distance-learning students. It has been used by approximately 1,500 students as an assessed coursework assignment across both aforementioned contexts. We have obtained sensitising findings to inform directions for our research from an initial run of Sleuth with our campus cohort

Overall, the students in the cohort performed to a very high standard with a median grade of 90.67% (Q1: 75.79 and Q3: 96.49). In taking module tests the same cohort performed considerably poorer with a median grade of 66.94% (Q1: 53.33% and Q3: 84.17%). However, in performing a

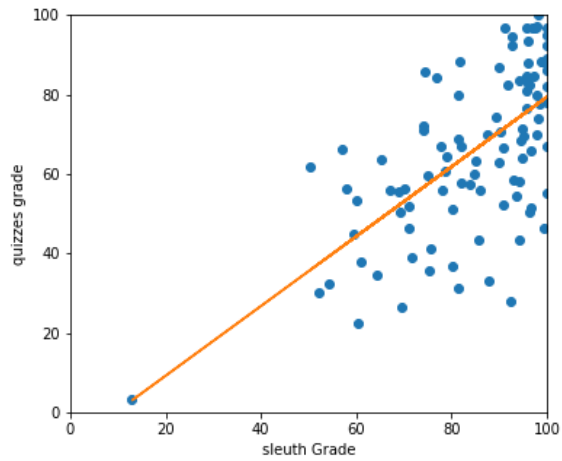

Fig 3: scatter plot of sleuth grades against test scores with regression

Spearman's rank correlation between the two assignments, a positive correlation was found ($r=0.63$ $p<0.01$). Whilst some of the stronger performances in Sleuth can be explained by attainment gains through coursework over test conditions, we believe that much of this can be attributed to improved student motivation through the aforementioned game design principles. This supposition is supported by the high levels of student activity from the cohort where there was an average of 158 attempts per student.

Such high levels of activity occasionally segued into undesirable, obsessive behaviours. For example, students demanding deadline extensions to achieve grades of 100%, students working on Sleuth throughout the night at the expense of class attendance, and students posting angrily to VLE forums targeting the fairness and accuracy of the autograder as they struggled to identify the correct solution. Contrary to our expectations, students made little strategic use of the open level design. Instead the majority progressed doggedly in sequence. A Spearman's rank correlation comparing the case and stage numbers of students attempts with timestamps revealed an ordinal tendency within the data, with a mean r of 0.947 across the cohort. This linear approach was often at the cost of many failed attempts at harder code crimes as they fixated on individual problems. Nevertheless, we found evidence of self-pacing as is shown by Fig 4 which presents changing grade distributions across the cohort for the period of the assessment.

In a post-assignment survey students reported on their perception of the difficulty of Sleuth scoring a mean rating of 2.7 (5-point Likert scale where 1 represents most difficult). 36% of students found the assignment to be "a little difficult" and 51% found it to be "just right". In open-form feedback students requested extra features such as hidden "Easter egg" levels and leaderboards.

VI. Planned Research

The cyclical nature and multiple deployments of Sleuth, combined with its multilevel design provides a unique opportunity for in the field testing with controls. Based on our preliminary findings and informed by the literature we envisage a number of investigations the first of which is already underway.

Our first investigation explores the role of feedback in student performance. Keuning et al identify that the majority of automated feedback from code exercises takes the form of

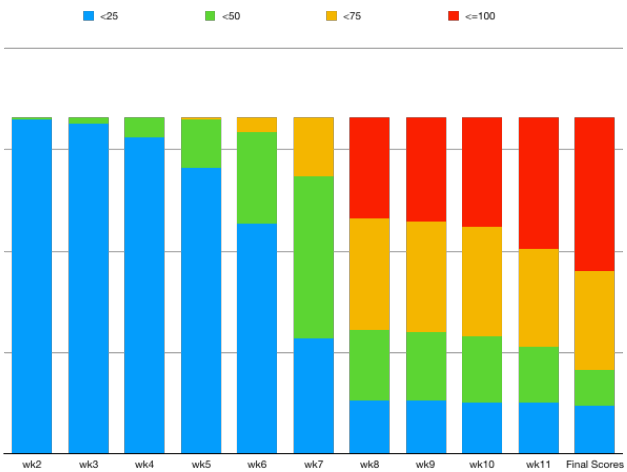

Fig 4: changing grade distributions across the period of the Sleuth assessment

'Knowledge about mistakes', and that this alone is not sufficient for students to make progress [12]. Using our preliminary data we have identified two cases which are particularly challenging for students. For one case students who submit incomplete or incorrect attempts are offered a chance to see an automatically generated solution in return for restarting their attempt with a new variation. For the other case students are repeatedly offered hints with each incorrect or incomplete submission. Testing is underway with our campus cohort of 280 students, whilst our online cohort of 580 students is acting as a control group. We will be measuring the degree to which the hints and solutions improved performance and reduced the number of attempts, and also measuring student preferences for these types of feedback.

Our second investigation seeks to mitigate the sequential approach of students in solving crimes as established by our preliminary data. We suspect that the layout of crimes and cases in a sequentially ordered grid heavily reinforces this behaviour. We will develop an alternative interface design which presents puzzles hidden in a 2D map of 'Console City,' the fictional setting of Sleuth. Students will navigate their way to available crimes in the mode of open world exploration games such as 'Grand Theft Auto '. Whilst encouraging students into a more flexible approach, such a design change also makes the presentation of student progress less transparent, so we will also be investigating how this affects student motivation.

The high levels of student achievement in the preliminary data combined with the student perception of task difficulty implies some degree of intrinsic motivation at play [5]. We propose to test how intrinsically motivated students are by providing two cohorts with "Easter egg" crimes which appear only once a certain score has been reached. These provide an opportunity to introduce 'Class 3' problems allowing for the application of alternative solution strategies [15]. In the control cohort these crimes will have summative marks attached to them, but in the experimental cohort there will be no extrinsic reward for solving the crimes.

The character of 'The Chief' provides other opportunities for investigation. We are interested in the effectiveness of encouraging student behaviours through platform interventions. One scenario where this might be applied is where students have repeated failed to solve a crime and are suspended from the case. In this circumstance we would attempt to stimulate continued engagement by 'The Chief' suggesting an alternative case to be attempted.

The discrepancy between Sleuth grades and test results in preliminary data calls us to question whether in its current form Sleuth is providing a suitably rigorous environment for summative assessment. We suspect that some weaker students may be solving crimes with an inappropriate degree of peer support. One possible solution would be to introduce a degree of repetition in the game. Students would be occasionally presented with crimes that they had already solved once, but now they would be challenged to solve them with a limited number of attempts. From this intervention we would be looking for a reduction in the discrepancy between the two assessments and an improvement in performance of the bottom quartile in the tests.

VII. Conclusion

We have presented Sleuth as an ideal platform for iterative in-the-field investigation into gamified learning and automated feedback. We have positioned this work within the fields of automated feedback and exercise generation in computing pedagogy and gamification within education. We have described the specification of Sleuth and its design motivations. We have explained our preliminary findings from use in the classroom and used these to set out a scheme of research.

References

[1] Y. Qian and J. Lehman, "Students' Misconceptions and Other Difficulties in Introductory Programming," ACM Transactions on Computing Education, vol. 18, no. 1. pp. 1–24, 2017.

[2] R. P. Medeiros, G. L. Ramalho, and T. P. Falcao, "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education," IEEE Transactions on Education, vol. 62, no. 2. pp. 77–90, 2019.

[3] S. Deterding, From Game Design Elements Tot Gamefulness: Defining "gamification." 2011.

[4] K. Seaborn and D. I. Fels, "Gamification in theory and action: A survey," International Journal of Human-Computer Studies, vol. 74. pp. 14–31, 2015.

[5] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," American Psychologist, vol. 55, no. 1. pp. 68–78, 2000.

[6] "Non-continuation: UK Performance Indicators 2017/18 | HESA." [Online]. Available: https://www.hesa.ac.uk/news/07-03-2019/non-continuation-tables. [Accessed: 05-Apr-2019].

[7] B. Marín, J. Frez, J. Cruz-Lemus, and M. Genero, "An Empirical Investigation on the Benefits of Gamification in Programming Courses," ACM Transactions on Computing Education, vol. 19, no. 1. pp. 1–22, 2018.

[8] L. E. Nacke and S. Deterding, "The maturing of gamification research," Computers in Human Behavior, vol. 71. pp. 450–454, 2017.

[9] J. Hamari, J. Koivisto, and H. Sarsa, "Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification," 2014 47th Hawaii International Conference on System Sciences. 2014.

[10] S. O'Donovan, J. Gain, and P. Marais, "A case study in the gamification of a university-level games development course," Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference on - SAICSIT '13. 2013.

[11] Dicheva D, Dichev C, Agre G, Angelova G, "Gamification in Education: A Systematic Mapping Study," Educational Technology & Society, vol. 18, no. 3, pp. 1176–3647, 2015.

[12] H. Keuning, J. Jeuring, and B. Heeren, "A Systematic Literature Review of Automated Feedback Generation for Programming Exercises," ACM Transactions on Computing Education, vol. 19, no. 1. pp. 1–43, 2018.

[13] S. Narciss, "Feedback strategies for interactive learning tasks," in Handbook of Research on Educational Communications and Technology., S. A. B. Kathy L. Schuh, Ed. Routeledge, 2008, pp. 125–144.

[14] Sadigh, D. Seshia, S A. , Gupta, M, "Automating Exercise Generation: A Step towards Meetingthe MOOC Challenge for Embedded Systems," in Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education - WESE '12, Tampere, Finland, 2013, pp. 1–8.

[15] N. Pinkwart and N.-T. Le, "Towards a classification for programming exercises," in Proceedings of the Workshop on AI-supported Education for Computer Science., 2014, pp. 51–60.

[16] CodeCombat, "CodeCombat: Learn to Code by Playing a Game," CodeCombat. [Online]. Available: http://codecombat.com. [Accessed: 23-Oct-2019].

[17] J. Bishop, R. Nigel Horspool, T. Xie, N. Tillmann, and J. de Halleux, "Code Hunt: Experience with Coding Contests at Scale," 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. 2015.