# The SOB Monitor: Supporting Competency-Based Assessment with Gamification

**David Dorvekinger**
Department of Computer Science, Middlesex University, London, UK

**Michael Heeney**
Department of Computer Science, Middlesex University, London, UK

**Kelly Androutsopoulos**
Department of Computer Science, Middlesex University, London, UK

## Abstract

The SOB Monitor is a web-based platform that supports competency-based assessment through Student Observable Behaviours (SOBs). SOBs are defined as measurable units aligned with learning outcomes and are assessed at Threshold, Typical, and Excellent levels. The assessment strategy involves continuous assessment with multiple resubmission opportunities, allowing for real-time monitoring of student progress. It is adopted on several undergraduate Computer Science modules, including first-year Programming, Systems and Architecture, Foundations of Computing, and Design and Development of Applications.

In this paper, we describe assessment with SOBs, the new architecture of the SOB Monitor platform, its implementation and how it supports competency-based assessment in Computer Science undergraduate modules. Gamification elements including leaderboard and rankings have been introduced to enhance student motivation, engagement and self-regulated learning within pass/fail modules.

**2012 ACM Subject Classification** Applied computing → Interactive learning environments

**Keywords and phrases** Competency-based education, interactive learning environments, assessment, higher education, computer science education, gamification

**Digital Object Identifier** 10.4230/OASIcs.ICPEC.2026.13

**Category** Short Paper

## 1 Introduction

There are a number of challenges when designing an undergraduate first-year Computer Science module. One such challenge is the wide variety of student prior knowledge, particularly in programming [4], where some students have never coded before, while others have for years. This leads to difficulties in pacing and structuring of material to ensure that all students are equally engaged. It can also lead to difficulties in designing assessment that is fair across diverse student backgrounds, as this can influence performance and progression, e.g. assessments that seem reasonable for a beginner may be trivial for an experienced student. Moreover, in programming, a correct program does not mean that the student has a deep understanding of the concepts [13]. A student might produce a working program by trial-and-error, external help or reusing existing code, while another student might submit a partially working program but have a stronger conceptual understanding.

These problems are not unique to just programming modules, but can take different forms in other modules such as Systems and Architecture. A key difficulty is again what counts as understanding. Students may be able to memorise how components such as caches, pipelines or memory hierarchies work and reproduce this in an exam, but still lack a coherent mental model of how those components interact at runtime [12]. With the growing availability


of generative artificial intelligence (GenAI) tools, students are regularly using these for generating solutions, raising further concerns about academic integrity, over-reliance from students, misleading outputs and other topics such as bias and vulnerabilities [6]

We adopt a competency-based assessment approach for alleviating these problems, by prioritising the demonstration of learner competence through measurable knowledge, skills, and professional behaviours rather than progression based on time or content coverage. This approach supports responsible use of GenAI tools. Students can use GenAI tools as part of their learning process, but for assessment are required to articulate and defend their work, ensuring that understanding is assessed. Synthesising perspectives from both conceptual and applied research, this approach is characterized by clearly defined competencies aligned with stakeholder and societal needs, flexible learning pathways that accommodate individual pacing and prior experience, and an emphasis on authentic, performance-based assessment [8].

The SOB Monitor is a web-based platform that supports competency-based assessment of first-year Computer Science modules through Student Observable Behaviours (SOBs) [1], which represent fine-grained decompositions of the learning outcomes. Assessment of SOBs in the context of first-year programming education, along with the lessons learned, is described in [2]. The SOB Monitor enables continuous monitoring of student progression, allowing for early intervention. The first-year modules are not graded, but pass/fail, and maintaining student motivation beyond minimum expectations can be difficult. Some modules include several group-based assessment activities, which complicates the management and marking of SOBs.

Module creation and maintenance can be difficult, particularly when troubleshooting login failures and tracking errors across module instances. As the SOB Monitor has grown in popularity across the institution, supporting 28 modules in Computer Science, Engineering, and Mathematics has further amplified these issues.

This paper addresses these limitations with the re-development and extension of the SOB Monitor to enhance platform security, enable support for multiple modules, and address limitations in the existing project infrastructure. The new architecture is presented which consists of a dedicated web framework (Django) with new code infrastructure that improves the platform's ease of use across multiple modules. Gamification elements have been introduced, specifically leaderboard and rankings, to enhance student motivation and engagement, particularly with more advanced material. Features for supporting group activities with multiple student and SOB marking have also been introduced.

## 2 Related Work

Competency-based education is a framework for teaching and assessment that shifts the focus from time-based progression to mastery of demonstrable competencies [15]. This approach supports self-paced learning and accommodates variation in student progression. By design, competency-based education is well suited to cohorts with diverse ability levels, as it allows learners to progress according to their individual development rather a fixed pace [8].

A number of widely used Learning Management Systems (LMSs) support competency-based education through assessment, outcome-alignment, and progress-tracking features. Platforms such as Moodle LMS (`https://moodle.com/products/lms/`), Blackboard LMS (`https://www.blackboard.com/`), Canvas LMS (`https://www.instructure.com/canvas`), and OpenOLAT (`https://www.openolat.com/`) include mechanisms for representing competencies, learning outcomes, goals, or related progress indicators. However, these tools are not built around competency-based education as a core pedagogical model, this capability can be added as a plug-in or third party tool in some cases, but it takes away from the coherent feel and adds complexity.

Unlike LMS, Artemis[1] implements competency-based adaptive learning more explicitly and focuses on CS education and programming-heavy education. It supports competency-based education, including competency definitions, competency relations, mastery tracking, and personalized learning paths. The evidence used for competency tracking is generated primarily through student interaction with the platform and the competency is inferred from this generated data. While Artemis supports manual assessment and tutor feedback on submissions, live oral explanations, "show me" demonstrations, and dialogue based tutor-student interactions do not appear to be central evidence sources in the native competency-tracking model [16]. This limits the system's ability to capture forms of understanding best assessed through discussion, observation, or oral defense. As a result, it may not fully capture the competencies that require explanation, judgment, reasoning, or observed performance.

Gamification uses game elements, such as leaderboards and rankings, to increase engagement [7]. It creates an excellent synergy for competency-based learning, as these elements encourage progress and achievement. These game elements give students a richer experience and push them to pursue more than they would otherwise [10]. In some cases these elements can introduce negative effects. For example, the competitive nature of the leaderboard can cause anxiety for some students, as they may feel they are falling behind their peers which in some cases can reduce motivation, instead of enhancing it [9]. This is particularly relevant in scenarios where students are not progressing at the same pace. These effects can be addressed through proper explanation and support from the teaching team. In some situations this anxiety can overshadow the student's understanding of actual deadlines or expectations. In these cases the teaching team plays an important role of giving reassurance, additional context, and encouragement, helping students to stay motivated, and bringing them back to a clearer understanding of their progress.

Gamification elements can be used in some LMS either with built-in features or using plug-ins. For example, in Moodle, features such as ranking, leveling up, badges etc. can be added to enhance engagement and learning [5]. In [5], the authors note that adding too many gamification features could lead to distracting a student.

## 3 Assessment with Student Observable Behaviours (SOBs)

The assessment strategy is designed to support both constructive alignment [3] and competency-based learning [15]. Learning outcomes are first defined, and the teaching and assessment activities are then aligned with these outcomes. Students participate in relevant, student-centered learning activities, developing their skills through active engagement while tutors act as coaches rather than providing step-by-step instruction [14]. The emphasis is placed on students demonstrating their learning through practical application rather than solely through theoretical understanding. The strategy also incorporates continuous assessment, multiple resubmission opportunities, and real-time monitoring of student progress.

**Student Observable Behaviours.** (SOBs) [1] describe fine-grained decompositions of the learning outcomes and are categorised into three levels. The Threshold level defines the required SOBs that students must demonstrate to pass the module. Typical and Excellent levels are provided to extend beyond the minimum requirements, thereby avoiding the risk of stunting knowledge acquisition, as it can be an issue in certain competency-based educational settings [17]. This multi-level structure enables competencies to be described more precisely

[1] `https://artemis.tum.de/`

across levels of achievement, thereby mitigating the tendency of competency-based approaches to become overly discrete or fragmented. By incorporating multiple levels of performance, the model captures a broader range of knowledge and skill development. Expectations are specified more clearly at each stage, supporting transparent and consistent assessment practices that reduces ambiguity in what is required.

The programming module has 33 SOBs: 11 Threshold, 15 Typical and 7 Excellent. The deadline for completing SOBs is the last day of the teaching term (a term consists of 12 weeks) but they can be demonstrated multiple times, anytime, until then. We have attached some suggested dates to the SOBs of when we believe students will be able to complete them by. Once the SOBs pass this date and have not been marked, they become overdue. At the start of the module, students know of all SOBs and learning materials. The approach to SOBs varies slightly between modules. For example, in the programming module [2], new weekly lab activities mapped to SOBs are provided for Threshold SOBs that must be completed during class time. Once a student completes an activity, they present it to the tutor, who asks short questions to check understanding. If the tutor is satisfied, the SOB is recorded as achieved on the SOB Monitor. Feedback is given regardless of the outcome. For example, for the Threshold SOB *"Show and explain a simple program using a for loop"*, the activity for this SOB could be "Write a Racket function that takes a list of numbers and returns their sum. E.g., (func '(2 4 3)) returns 9. Make sure to test your function and explain how it works.". The student will present their solution to the tutor, who will ask questions like "how does the for-loop work?", "what happens if the list is empty?" etc. If the student's solution is correct (or with some minor changes that the student can determine through a discussion with the tutor) and shows the understanding of what has been done, the SOB will be marked. The student could also get this SOB if they implemented a for loop as part of a project or another SOB, i.e. Typical or Excellent. Some higher-level SOBs subsume lower-level ones. For example, the Typical SOB *"Write a function that uses vectors to solve a simple problem"*. If the student implements a function for this that uses a for loop, the student is able to get the for loop SOB as well.

## 4 The SOB Monitor

The SOB Monitor supports both students and teaching staff, with their own dedicated views. Students can track their own progress and understand how they compare within their cohort, while preserving the privacy of other students' data (see Figure 1). Leaderboard and ranking elements have been added in the student view. The rank is calculated using the number of SOBs a student has achieved, comparing that count to other active students and their achieved SOB count in the same cohort. Using this rank, the percentile is calculated by subtracting the student's rank from the total number of active students in the cohort, dividing this result by the total number of active students, and multiplying by 100. If multiple students have the same count of achieved SOBs, they will have the same rank.

Staff can monitor all enrolled students via a detailed table of SOB achievements by level (see Figure 2), access cohort performance reports, and manage SOBs by creating, editing, and annotating them with dates and notes. The staff interface includes a dashboard for summarising student progress against module expectations, an administration page for managing staff roles and permissions, a SOBs view for creating and editing SOB details and timings, and a monitoring view for tracking student progress and providing comments. As a new feature, students can be organised into groups for project work that can be jointly marked. In the assessment view, staff can observe SOBs, lock submissions in cases of plagiarism, or

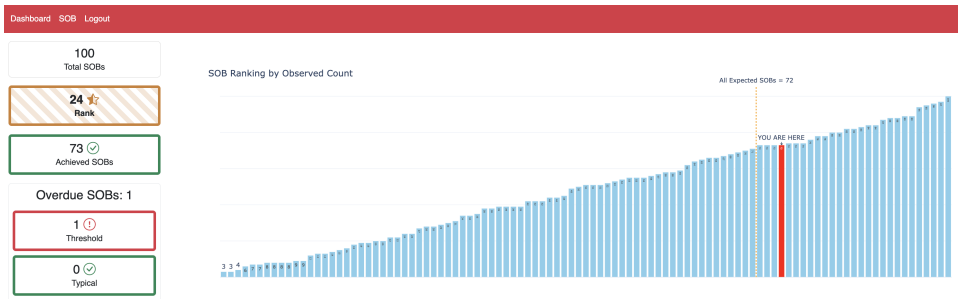


**Figure 1** Student Dashboard for SOB Monitor, showing student individual ranking compared to other students, SOBs completed, overdue SOBs and total available.

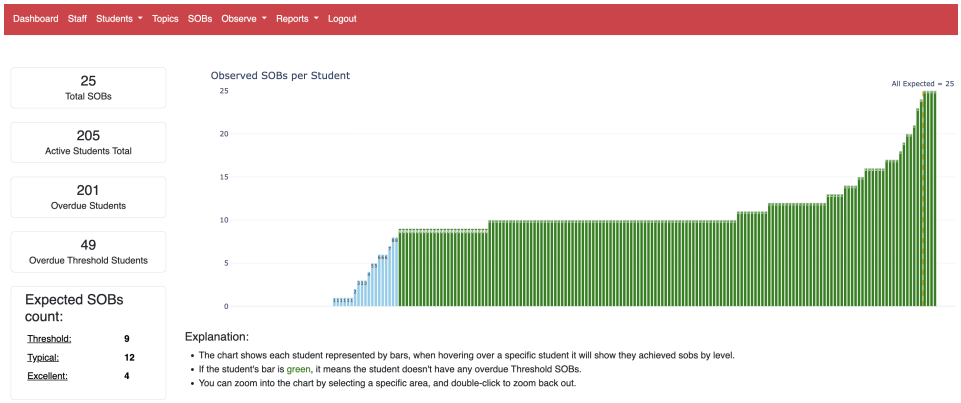


**Figure 2** Staff dashboard showing each student as an individual bar, with an expectation marker and colour coding to indicate progress.

assign "in-progress" attempts to avoid duplication (i.e. showing the same SOB to another tutor). The reports view provides analytics on progression, pass rates, overdue SOBs, and other key metrics.

## 4.1 Architecture and Implementation

The architecture of the SOB Monitor is illustrated in Figure 3. Instead of extending the previous PHP codebase, the platform has been rebuilt around a Python based web framework, Django, which provides a clearer structure for request handling, application logic, database interaction, authentication, and page rendering. This creates a more consistent codebase and simplifies future development as the platform expands to support additional modules. The Django application is served through Gunicorn, with both components packaged together inside a Docker container, while Nginx and PostgreSQL remain installed directly on the host machine. The deployment infrastructure has also been redesigned through a Continuous Integration and Continuous Deployment pipeline that builds and reloads the Docker container from the latest GitLab version. This reduces manual deployment work and allows changes to be released more consistently.

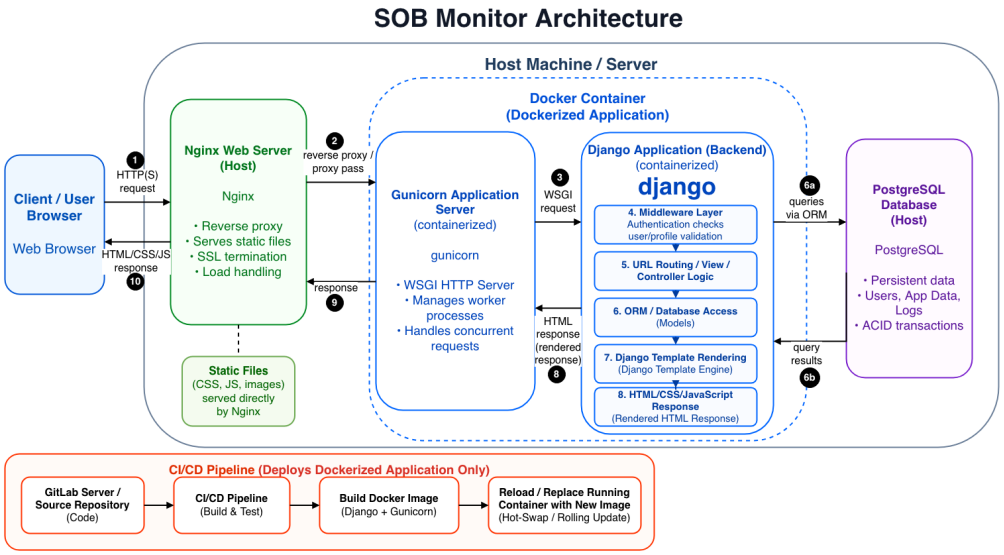


**Figure 3** System Architecture of SOB Monitor as described above.

## 5 Observations

We have observed that students like being able to work at their own pace, deciding when they are ready to complete their SOBs, and fitting assessment around their other commitments, e.g. part-time work or caring responsibilities. The SOB Monitor allows students to view their progress regularly and help manage SOB completion. The suggested dates given with SOBs, help students to not fall too behind. We have observed students working harder around the suggested dates, or attending extra, non-timetabled sessions.

Since adding the leaderboard and rankings, students are discussing their progress actively and asking others how they are demonstrating work, if they can practice submissions together and asking for feedback when necessary. We have also noticed some healthy competition, where students are trying to be ranked number 1, many aiming to complete all possible SOBs. When a student passes Threshold and/or all SOBs, the leaderboard is animated with celebrations for the student. A drawback of the leaderboard is that some students who need more time to complete SOBs may feel rushed into showing work to tutors when it is not ready or not their own (e.g. generated by GenAI tools).

To assess the effect of the gamification elements, we compared three Programming cohorts: the September 2024 cohort, which used the SOB Monitor without leaderboard and ranking features, and the January 2025 and September 2025 cohorts, which used the gamified version. SOB completion was normalised by cohort size to account for differences in enrolment. The chart in Figure 4 shows the percentage of each cohort that completed each Programming SOB, with SOB numbers 1–33 shown on the x-axis. These SOBs are organised by level: SOBs 1–11 are Threshold, SOBs 12–26 are Typical, and SOBs 27–33 are Excellent. Across the 33 Programming SOBs, the cohorts using the gamified SOB Monitor showed a broadly comparable or stronger pattern of completion, particularly across the Threshold SOBs, where a high proportion of students completed the required behaviours. Completion of Typical and Excellent SOBs remained lower overall, which is expected because these levels extend beyond the minimum pass requirement. However, the presence of visible rankings and progress comparison appears to support continued engagement beyond the Threshold level for at least

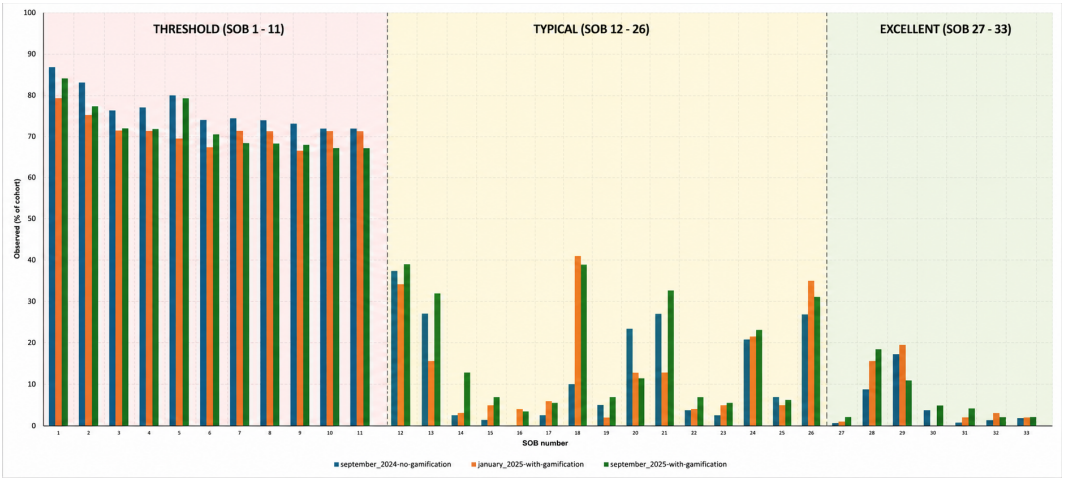


**Figure 4** Normalised Programming SOB completion by cohort. SOBs 1–11 are Threshold, SOBs 12–26 are Typical, and SOBs 27–33 are Excellent.

some students. Some Typical SOBs were attempted by very few students in the September 2024 cohort, including SOBs 14, 16, and 18. These results should not be interpreted as causal evidence, as the cohorts differ by intake and timing, but they provide useful early evidence that gamification may contribute positively to student engagement with competency-based assessment.

In Programming, new Threshold SOB questions are introduced in each class and completed in-session by students who have engaged with the exercises, preventing reliance on SOBs as a substitute for learning. In Design and Development of Applications, students work in groups on projects, with SOBs assessed using the SOB Monitor's group assessment feature, making marking more efficient. The SOB Monitor scales effectively to large cohorts, such as the Foundations of Computing module (300+ students), where a fixed set of SOB activities is released for students to complete independently and present when ready for tutor review. In Systems and Architecture, students typically work in pairs to complete SOBs using methods such as physical circuits, digital twins [11], or discussion. Progression through workbook material is required beforehand, and demonstrated understanding in discussion may be sufficient for awarding a SOB. Across all modules, tutors confirm attainment through observation and targeted questioning, with SOBs recorded in the Monitor.

## 6 Conclusion and Future Work

We have presented the SOB Monitor platform that supports competency-based assessment using SOBs within, but not limited to Computer Science. This new implementation enhances platform security, enables support for multiple modules, and includes new features for supporting the assessment of group work. In addition, gamification elements have been implemented to enhance student engagement and motivation. Initial observations have been promising. We plan to conduct further studies to evaluate the effectiveness of the gamification elements that were introduced. We also plan to extend the SOB Monitor so students can submit code directly and run it against tutor-defined test cases. The resulting submissions and progress data, such as the number of test cases passed, could help track student development over time and support tutors in improving teaching materials.

## References


1 K. Androutsopoulos, N. Gorogiannis, M. Loomes, M. Margolis, G. Primiero, F. Raimondi, P. Varsani, N. Weldin, and A. Zivanovic. A racket-based robot to teach first-year computer science. In *7th European Lisp Symposium*, volume 54, pages 54–61, Paris, France, May 2014.

2 K Androutsopoulos, M Heeney, S Smith, and M A Ticar. Lessons from adopting a competency-based assessment approach for an introductory programming module. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2*, pages 743–744, 2025.

3 J. B. Biggs and C. Tang. *Teaching for quality learning at university.* Open University Press., Buckingham, England, 4th ed. edition, 2011.

4 N C. C. Brown, S Sentance, T Crick, and S Humphreys. Restart: The resurgence of computer science in uk schools. *ACM Trans. Comput. Educ.*, 14(2), 2014. doi:10.1145/2602484.

5 Claudia de Armas de Armas, Isaac Guillermo Gonzales Vizcarra, Douglas Lima Dantas, Sergio Takeo Kofuji, and Antonio Carlos Seabra. Analysis of gamification elements in the virtual learning environment context. In *2019 IEEE World Conference on Engineering Education (EDUNINE)*, pages 1–5, 2019. doi:10.1109/EDUNINE.2019.8875851.

6 P Denny, J Prather, B A Becker, J Finnie-Ansley, A Hellas, J Leinonen, A Luxton-Reilly, B N Reeves, E A Santos, and S Sarsa. Computing education in the era of generative ai. *Communications of the ACM*, 67(2):56–67, 2024. doi:10.1145/3624720.

7 S Deterding, D Dixon, R Khaled, and L Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. Association for Computing Machinery.

8 J Frank, L Snell, O ten Cate, E Holmboe, C Carraccio, S Swing, P Harris, N Glasgow, Dr. C Campbell, D Dath, R Harden, W Iobst, D Long, R Mungroo, D Richardson, J Sherbino, I Silver, S Taber, M Talbot, and K Harris. Competency-based medical education: Theory to practice. *Medical teacher*, 32:638–45, August 2010. doi:10.3109/0142159X.2010.501190.

9 Michael D. H and Jesse F. Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & Education*, 80:152–161, 2015. doi:10.1016/j.compedu.2014.08.019.

10 Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work? – A literature review of empirical studies on gamification. In *2014 47th Hawaii International Conference on System Sciences*, pages 3025–3034, 2014. doi:10.1109/HICSS.2014.377.

11 M Heeney, K Androutsopoulos, and F Raimondi. Implementing a digital twin for a robotic platform to support large-scale coding classes. In *5th International Computer Programming Education Conference*, 2024. doi:10.4230/OASIcs.ICPEC.2024.15.

12 G L. Herman, M C. Loui, and C Zilles. Students' misconceptions about medium-scale integrated circuits. *IEEE Transactions on Education*, 54(4):637–645, 2011. doi:10.1109/TE.2011.2104361.

13 R Lister, E S. Adams, S Fitzgerald, W Fone, J Hamer, M Lindholm, R McCartney, J E Moström, K Sanders, O Seppälä, B Simon, and L Thomas. A multi-national study of reading and tracing skills in novice programmers. In *ITiCSE: Innovation and Technology in Computer Science Education*, pages 119–150, New York, NY, USA, 2004. ACM. doi:10.1145/1044550.1041673.

14 Seymour Papert and Idit Harel. *Constructionism.* Ablex Publishing Corporation, Norwood, NJ, 1991.

15 Joze Rugelj, Sven Knockaert, Roel Van Steenberghe, Luk Schoofs, Janne Salonen, Kari Björn, Jose L. Marzo, and Carlos Vaz de Carvalho. Competence based joint study program on advanced networking technologies for blended learning. In *Proceedings of the 9th International Conference on Information Technology Based Higher Education and Training.* IEEE Press, 2010.

16 Maximilian Sölch, Moritz Aberle, and Stephan Krusche. Integrating competency-based education in interactive learning systems. *arXiv preprint arXiv:2309.12343*, 2023. doi:10.48550/arXiv.2309.12343.

17 L Wheelahan. How competency-based training locks the working class out of powerful knowledge: A modified bernsteinian analysis. *British Journal of Sociology of Education*, 28:637–651, September 2007. doi:10.1080/01425690701505540.