Community-driven Course and Tool Development for CS1

Boyd Anderson*
boyd@comp.nus.edu.sg
National University of Singapore
Singapore

Martin Henz*
henz@comp.nus.edu.sg
National University of Singapore
Singapore

Kok-Lim Low*
lowkl@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

In 2012, the authors took responsibility for a CS1 course with 45 students. This experience report reviews the subsequent 10-year learning process of engaging undergraduate students to facilitate small-group teaching and to design and develop an online learning environment to conduct what became our university's flagship CS1 course, currently enrolling 749 students. The course inherited an emphasis on small-group learning from its role model, MIT's 6.001. The size of the learning groups is limited to eight students per group, which currently requires a team of 105 student facilitators. The resulting need for student engagement and scaling motivated the development of a new web-based programming environment and assessment management system custom-made for the course. The system was conceived, designed, and implemented by students of the course, which provided the glue for building a sustainable and scalable community of learners, educators, and student software developers. This experience report describes the pedagogic approach, the course structure, and software system to accommodate the needs of this community. A qualitative and quantitative analysis of the impact of the course over the last four years provides evidence for its efficacy. We hope that this report serves as inspiration for similar large-scale pedagogic efforts that bring learners, educators, and student developers together to form sustainable and scalable learning communities.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**; • **Social and professional topics** → **Computational thinking**.

KEYWORDS

learning management system for programming, introductory programming, structure and interpretation of computer programs

ACM Reference Format:
Boyd Anderson, Martin Henz, and Kok-Lim Low. 2023. Community-driven Course and Tool Development for CS1. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569740

*All authors contributed equally to this research.

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569740

1 BACKGROUND

CS1101S was established at the National University of Singapore (NUS) in 1997 as a more challenging and exciting alternative to the regular CS1 course, which followed a traditional curriculum centered around C-programming. It enrolled 30 to 45 students each year, who opted out of the latter to join CS1101S after passing a selection process. The course used the textbook "Structure and Interpretation of Computer Programs" (SICP) by Harold Abelson and Gerald Jay Sussman [1] and was modeled after MIT's 6.001, which was conceived by Abelson and Sussman. Following the publication of the second edition of SICP in 1996, many leading universities similarly adopted the SICP approach to CS1. CS1101S at NUS inherited 6.001's rigorous approach and its use of the programming language Scheme for all programming examples and assessments. The most advanced students typically joined the team of facilitators in the following year, each in charge of a group of at most eight first-semester students. They developed homework assignments for their juniors and the necessary software support, for example involving entry-level algorithms and cryptography, and received modest monetary compensation for their efforts.

With MIT discontinuing 6.001 in 2008 and other leading universities changing their courses away from SICP and Scheme in the following years, a reassessment of CS1101S became necessary. The support among the faculty for an entry-level course with an emphasis on functional programming was strong enough to continue the SICP-based approach at NUS, but the support for Scheme as its programming language was waning. The second author decided to adapt the course to use JavaScript instead of Scheme, based on a positive experience in a programming course for non-computer-science majors, and based on an effort to adapt selected sections of SICP to JavaScript that started in 2008. The first enrolment after adaptation to JavaScript in 2012 comprised 45 students, handled by one instructor and six facilitators.

While the selection process was gradually removed, the course grew to 120 students and 15 student facilitators by 2017, still comprising only students who opted out of the regular C-based CS1 course. Three developments necessitated a revision in the course material, tools, and management in 2018: (1) Our computer science department decided to make CS1101S the required CS1 course for all students in its 4-year computer science undergraduate program, which meant that its enrolment was projected to more than triple from 2017 to 2018. (2) Student interest in computer science increased dramatically, which led to predictions of continuous post-2018 enrolment growth without lowering program admission criteria. (3) JavaScript had evolved in 2015 in several crucial ways, which made it possible to aim for a much closer textbook adaptation.

Continuing the established pattern of student-led tool development, the academic team engaged a prominent graduating CS

student, Evan Sebastian, to architect a new web-based learning environment. A team of five undergraduate students, mostly former CS1101S facilitators, designed and implemented a basic first version of the Source Academy (version "Cadet") in the mid-2018 break under Sebastian's guidance. In Semester 1 of Academic Year (AY) 2018/19, the initial version handled 420 students and 55 facilitators. Versions "Cadet" and "Knight" handled gradually increasing student numbers from AY 2019/20 to AY 2021/22, and the latest stable version "Rook" is serving 749 students and 105 student facilitators in Semester 1 of AY 2022/23. (CS1101S is also offered in Semester 2 of each AY, but enrollment is much smaller at around 25 students; we ignore the Semester 2 version in this report for simplicity.)

Section 2 recounts how the course evolved from MIT's 6.001 as a result of these scaling needs and the switch to JavaScript, and highlights core functional requirements for Source Academy. Section 3 describes the overall system design and architecture, while Sections 4 and 5 focus on the programming environment and learning management aspects of the system. Section 6 outlines the features of Source Academy that aim at improving student engagement. Section 7 provides quantitative and qualitative results spanning the last four years of CS1101S, and Section 8 discusses how educators may strengthen their own communities of learners for CS1 and beyond by engaging student developers and facilitators.

2 PEDAGOGIC REQUIREMENTS

SICP establishes a series of mental models for computation. Its first two editions use the language Scheme in their program examples, whose minimalistic, expression-oriented syntax allows the book to focus on the underlying ideas rather than the design of the chosen language. Aiming to retain SICP's pedagogy and generality of learning objectives, the JavaScript edition SICP JS [3] modernizes the material by covering the structure and interpretation of programs written in statement-oriented languages that include return statements, a language feature present in most widely-used programming languages today. Apart from pedagogic reasons outlined in [4] for choosing the statement-oriented language JavaScript for the program examples, a practical advantage was that the programs can be run easily on any device with a JavaScript-enabled browser, which provided the opportunity for Source Academy to become web-based. A web-based learning environment would simplify the scaling of the course by removing the need for learners to install any software apart from their web browser and facilitate the dissemination of system improvements, which we anticipated to be frequent, especially in the early phases of the development.

The value of a textbook lies in establishing a common language of discourse among students, facilitators, and staff and thus making scaling easier. To maximize the utilization of SICP JS, an integration of an online interactive edition of the book into Source Academy seemed desirable. The first two mental models of computation in SICP (JS) are the substitution model for evaluating functional programs outlined in SICP (JS) Chapter 1 and the environment model for execution of imperative programs in the presence of lexical scoping and higher-order functions described in Chapter 3 and implemented in the meta-circular evaluator of Chapter 4. A full comprehension of these mental models poses considerable challenges for a typical learner, which affects the scalability of the course. Interactive and graphical visualization of these models in Source Academy would help learners, facilitators, and academic staff to establish these mental models more reliably in large classes.

SICP (JS) often uses examples from mathematics to illustrate programming abstractions such as higher-order functions and streams. To engage students better, Source Academy should make good use of the media-rich computing environment of the web, including 3-D graphics, audio and video processing, and such components should be added to the system only when needed.

The sublanguages approach to teaching CS1 was pioneered for PL/I [14], and was widely adopted since then, see also [10] and [15]. Following this approach, Source Academy should restrict learners to sublanguages of JavaScript that contain just enough features for supporting the respective chapters of SICP JS to facilitate the scaling of the course to learners with diverse prior knowledge and exposure to programming in general and to JavaScript in particular.

A web-based environment to support the teaching of CS1101S provides a range of opportunities to improve student engagement using established gamification techniques [18], such as achievements, experience points, badges, and contests. A web-based system might include a semester-long game to provide the context for meaningful exercises and make it easy to engage students in media-rich programming contests.

Assessment is a potential obstacle to scalability. To give effective feedback and guidance, CS1101S comprises three sit-in written assessments in addition to the final assessment, 20 graded homework assignments, a sit-in programming assessment, and two oral assessments conducted by the student facilitator. Source Academy should therefore provide extensive support for the management of programming assessment through assessment management, automated grading, and grading monitoring features.

To illustrate the use of imperative programming, CS1101S includes a robotics component, based on LEGO MINDSTORMS [16]. A web-based system for CS1101S should allow students to write their robot programs in JavaScript and run them on their robots from the convenience of their browsers.

While Source Academy should avoid duplicating typical features of learning management systems, a bespoke system for teaching CS1 should provide features specific to programming, such as manual review and grading by facilitators, supported by automatic grading based on test cases.

3 SYSTEM DESIGN AND ARCHITECTURE

The requirements mentioned in the previous section necessitated a scalable backend and the adoption of a JavaScript library for the frontend that facilitates student developer recruitment and onboarding. Sebastian chose the React [17] library for the frontend, Elixir [8] for the backend, and Elixir's web server Phoenix [7].

The system evolved to combine three components: (1) an interactive web-based edition of SICP JS, (2) a web-based programming environment for learning how to write programs in custom-designed JavaScript sublanguages, and (3) a web-based learning management system for staff to set programming exercises, for students to solve and submit their solutions, and for facilitators to grade the submissions automatically or manually. The first two components are of interest to readers of SICP JS and casual learners who don't

require password protection. Source Academy therefore provides a server-less public version that is deployed to a `.org` domain from the GitHub repositories of Source Academy [20]. The third component is available in an enhanced version of Source Academy called Source Academy @ X and is hosted by a commercial cloud provider.

The textbook sources of SICP and SICP JS are subject to a Creative Commons license and available in a separate GitHub repository [23], which allows for generating a PDF version of SICP JS for printing, an interactive online version of SICP JS, and a comparison edition [2] that displays both books side-by-side, all from a single XML sources.

The programming-language-specific features of the programming environment are provided by a JavaScript package [19] loaded by the React frontend at build time to ensure separation of concerns, to facilitate testing, and to provide the option of supporting languages other than JavaScript in the future.

The implementations of the JavaScript sublanguages are sandboxed such that programs written by learners can only interact with the frontend through well-defined pathways. Students and staff can safely share URL-encoded programs that link to Source Academy containing the program ready to run. Learner programs can import media-rich extensions (see Section 4) by using JavaScript's `import` syntax. The extensions are dynamically loaded into the frontend from a trusted module repository [22].

4 WEB-BASED PROGRAMMING ENVIRONMENT

The web-based programming environment is available in both the public and enhanced versions of Source Academy, and specialized versions are also used in the interactive textbook in both versions and in the assessment management and grading components of the enhanced version, see Section 5.

Sublanguages

To realize the sublanguages approach described in Section 2, the student developers designed and documented [21] a series of JavaScript sublanguages. The frontend parses the program entered by the learner using the JavaScript package `estree` [9]. The parse tree is then checked to ensure it complies with the selected sublanguage. Before evaluation, the program undergoes a transpilation step to generate learner-friendly error messages and to ensure proper tail calls (PTC) even if the browser does not comply with the PTC requirement of the JavaScript specification. For details of the parsing and transpilation, see [4].

A decisive advantage of the sublanguages approach for community-based development is that custom-designed tools can be implemented in student term projects. The development of a web-based educational programming environment for the full language JavaScript would by far exceed the constraints of term projects.

Tools for mental models

Interactive and visual representations of the two main mental models of SICP (JS) enabled the scaling of CS1101S. Several successive student project teams designed and implemented suitable tools for these models. For the substitution model of SICP JS Chapter 1, the teams developed a stepper tool and integrated it in the program development component of Source Academy, as described in [13]. For the environment model of SICP JS Chapter 3, the teams developed an environment model visualizer, as described in [6].

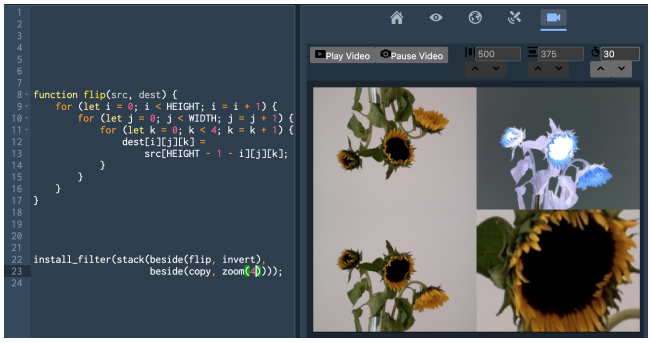

Figure 1: Imperative programming with video filters

Media

To take advantage of the today's media-rich world of web programming for inspiring programming exercises, student project teams developed dynamically-loaded modules for audio processing (for details see [12]), 2-D graphics as covered in SICP Section 2.2.4 (following [11]), visualization of functions with 2-D and 3-D codomains, constructive solid geometry (CSG), and video processing. Figure 1 shows a learning session in Source Academy that uses video filters to illustrate imperative programming using loops and suitable abstractions for combining such filters.

Robotics

To support the robotics component of CS1101S, several subsequent student team projects designed a virtual machine (VM) and a compiler of the SICP-Chapter-3 JavaScript sublanguage to the instruction set of the VM. The compiler is integrated in the web-based frontend such that a suitably configured and connected robot can execute the learner program entered in the browser by running the compiled code in the VM, which is written in C. For details, see [5].

5 LEARNING MANAGEMENT SYSTEM

Universities usually provide learning management systems (LMSs) to their students and staff to organize the courses that are relevant to them. CS1101S makes use of NUS's LMS for typical management functions such disseminating lecture material and calculating grades. Source Academy focuses on features we found useful but are not provided by commercial LMSs.

Assessment management and grading

Source Academy contains a management component for assessments that allows staff to upload assessments in XML and configure their opening and closing times. Programming assessments provide opportunities for automation by running student submissions on preconfigured test cases. Facilitators save time by relying on the autograding results to establish the correctness of the submission and can focus on assessing the achievement of learning outcomes.

Figure 2 shows the grading interface that allows facilitators to view and run student submissions, see the results of automatic grading, adjust the grade, and enter qualitative feedback. Automatic

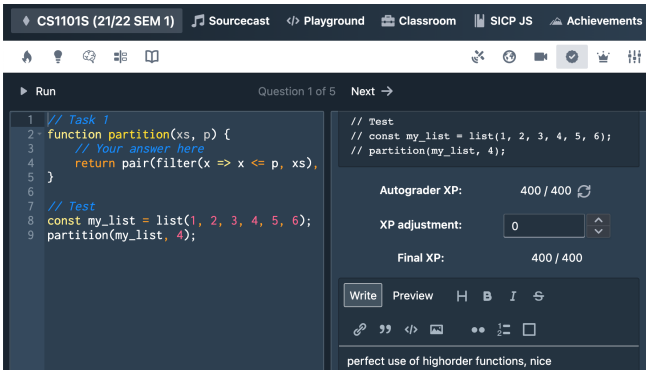

Figure 2: Grading interface for facilitators

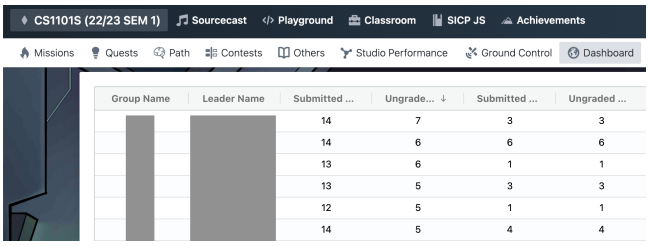

Figure 3: Dashboard with pivot table of facilitators who have difficulties giving timely feedback

grading is also employed in the practical assessment of CS1101S mentioned in Section 2.

Facilitator monitoring

The pedagogy literature provides ample evidence that timely feedback is a crucial component of effective teaching [24]. The facilitators participate in an orientation session at the beginning of the course and get detailed instructions on how to give feedback on assessments. Given the scale of CS1101S and the busy schedule of many student facilitators, it is inevitable that facilitators occasionally fall behind in their grading commitments. Figure 3 shows the "dashboard" of Source Academy, a pivot table of the status of assessment grading that allows academic staff to identify facilitators who need reminders of the importance of timely feedback.

6 IMPROVING STUDENT ENGAGEMENT

The student developers of Source Academy devoted considerable effort towards involving their juniors in activities that are broadly related to the course and that increase the enjoyment of the material and thereby enhance motivation and engagement.

Game

Students of CS1101S designed a visual-novel-style game that provides the context for all homework assignments of the course. The game follows a coherent semester-long plot to which the homework assessments refer. Provided that students can muster sufficient suspension of disbelief, they derive an additional sense of meaning, urgency, and accomplishment from their homework. The game of

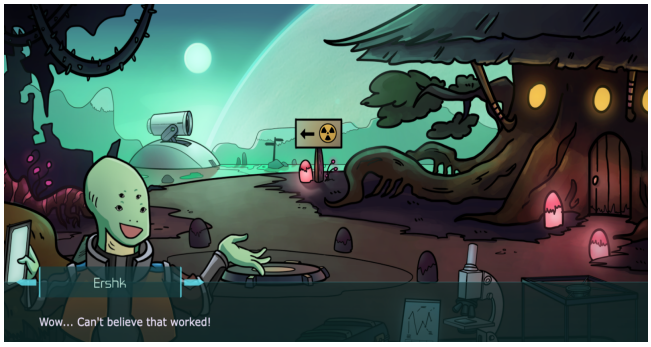

Figure 4: Typical scene from the CS1101S game. The planet's telescope is broken. The student needs to program a robot to repair it to capture the coordinates of a hostile entity...

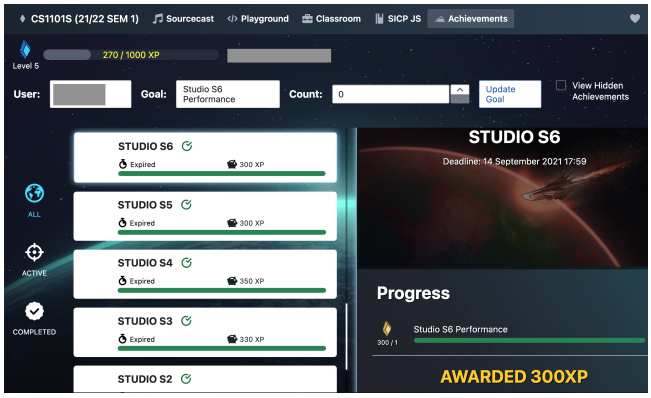

Figure 5: Display of student achievements

Source Academy underwent four major iterations and today features conditional control of game chapters, forks and loops in the plot, and extensive support for developing and testing the game plot and assets. Figure 4 shows a typical scene in the current plot that starts on board a spaceship and involves adventures on an alien planet. All game assets (scenes, avatars, collectibles, etc) are original and student-generated.

Achievements

Being able to review the achievements already attained increases students' confidence. Figure 5 shows the achievements component with typical gamification elements such as the student's level and experience points in the course, currently open assessments, badges, etc. Staff can use the achievements view to review a student's performance in the course.

Contests

To provide outlets for student creativity and the occasional comical relief, the course contains four programming contests, following the major groups of homework assessments. Students use Source Academy to develop their contest entries, and to cast their votes for the most popular entries. Typically, the criteria for winning a contest include popularity and program brevity. Source Academy has extensive support for managing and judging such contests.

7 RESULTS

NUS solicits anonymous student feedback on each course after the lecture period and before the final assessment. The quantitative and qualitative results reported in this section are based on the student feedback collected on CS1101S. The colored horizontal bar charts display the most recent course feedback, collected in AY 2021/22, with 670 invited students of which 580 (87%) responded.

Course

Figure 6 shows the overall student opinion on CS1101S from 2018 to 2021. Note that the feedback scores increased during the pandemic, which is a testament to the course's capability of online engagement, and that the feedback scores increased along with the number of students, which shows that the course managed to scale well during this period. Figure 7 shows the most recent overall opinion in more detail, and Figure 8 shows that the course is perceived as relatively difficult, indicating a sufficient engagement and challenge.

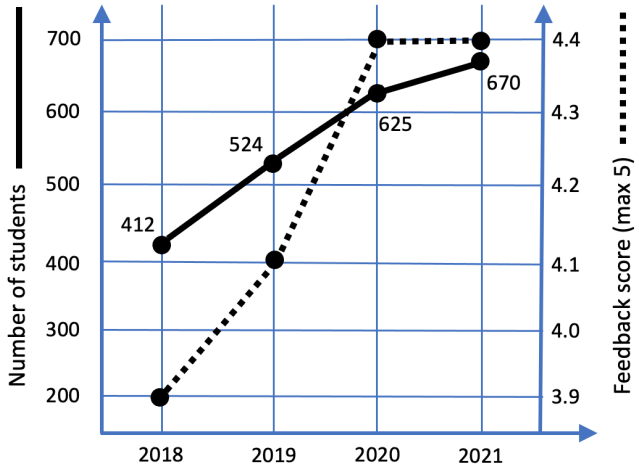

Figure 6: Student numbers and feedback scores ("What is your overall opinion of [the course]?": 1—Very poor, 2—Poor, 3—Average, 4—Good, 5—Very good. Response rates: 2018—74%, 2019—73%, 2020—77%, 2021—87%

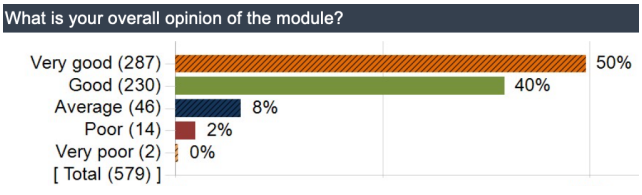

Figure 7: Student feedback on CS1101S in AY 2021/22

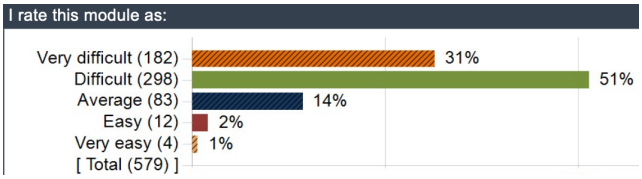

Figure 8: Perceived course difficulty in AY 2021/22

Textbook

The online edition of SICP JS was an independent student project by Samuel Fang, not connected to any academic credit or compensation. The project highlights the strength of the CS1101S community as Fang invested countless hours to build a system that serves their juniors. Fang won NUS's people's choice award for the most popular e-book in 2021. Reflecting on the award, STUDENT Fang remarked: *"I am happy knowing that the interactive textbook is getting the attention it deserves. I hope that the interactive SICP JS can continue to benefit students for years to come!"* The student feedback on the use of the textbook is encouraging, see Figure 9. Over the past 10 years, many dozens of students contributed to the GitHub repository of SICP JS by reporting bugs in the textbook or contributing exercise solutions, often using pull requests, and many more students have casually viewed the repository, and been exposed to online publishing such as single-source of truth and version control.

System

The students perceive Source Academy to be contributing to their learning, as evident from Figure 10. Figure 11 shows positive feedback on the game, overwhelming support for the achievement system, and encouraging feedback on the robotics component, despite the limitations imposed by the pandemic.

In the qualitative feedback on the course in 2021, 41 students mentioned Source Academy positively in response to the question "What I liked about the [course]". Two students mentioned Source Academy in response to the question "What I did not like about the [course]". The positive mentions included the following statements:

- *Source Academy is a very good environment to learn coding in as we are limited in what we can use*
- *Source Academy was a brilliant and fun platform to use.*
- *I'm really into Source Academy. This platform was incredible. It motivated me a lot to complete my tasks*

In addition to this recognition by their peers, the student development team won NUS's Outstanding Undergraduate Researcher Prize 2021/22 (Group category) and the Annual Digital Education Award 2021 (Team category).

The project course on Source Academy offered to first-year students in their second semester included 19 students in AY 2021/22,

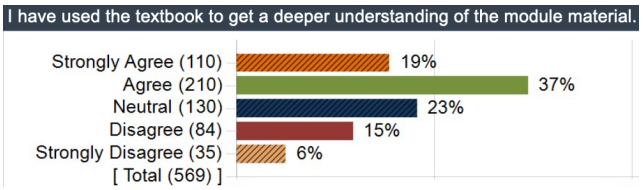

Figure 9: Student feedback on textbook use, AY 2021/22

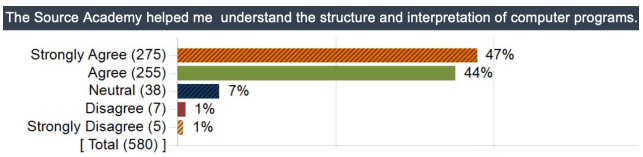

Figure 10: Student feedback on overall system efficacy

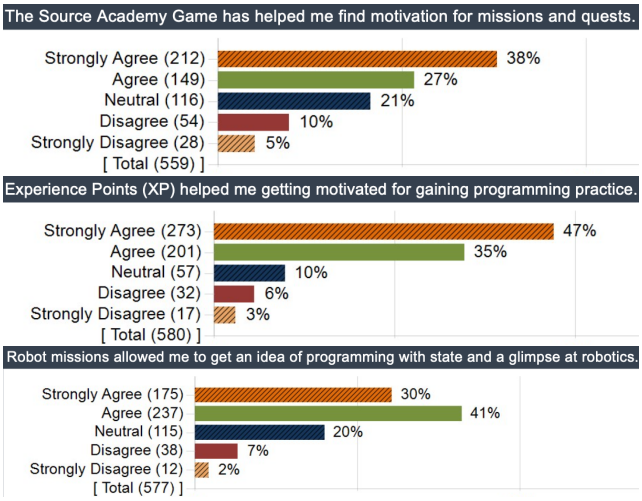

Figure 11: Student feedback on game component, achievement system, and robotics component

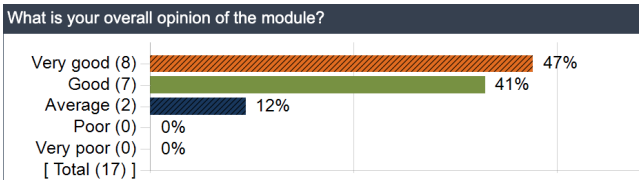

Figure 12: Student feedback on second-semester project course on Source Academy

out of which 17 (89%) responded to the feedback survey. Figure 12 shows that the students have a very high overall opinion of the project course. Their qualitative feedback included the following statements on "What I liked about the [course]".

- *Enhancing Source Academy for future students, trying new things, working alongside others doing similar work.*
- *It was an excellent opportunity to explore a codebase that was well–documented and well–organized, and to add a non–trivial feature extension to a platform that will be used by many of the future students in [NUS]...*
- *I enjoyed seeing the ideas that we came up with at the start of the semester becoming implemented and usable...*

For first-semester students, pedagogic value lies in using a student-built system, even if they don't become involved in the development, because they naturally empathize with the developers, get casually exposed to real-world software development using issue tracking and bug reporting, and explore the sources of Source Academy.

8 DISCUSSION

The reported results stem from several positive feedback loops. The student projects contributing to Source Academy supported CS1101S by providing a range of custom-built tools that improve the teaching. CS1101S improved the quality of the software projects that contribute to Source Academy by providing strong motivation for the participating student developers. The adaptation of SICP to JavaScript supported CS1101S by modernizing the course content. CS1101S supported the textbook project with volunteers for the development of online and interactive versions and with many textbook bug fixes and exercise solutions from the community of students. Each of these relationships strengthened the community of learners, educators, and student developers, which facilitated the recruitment of the next generation of student facilitators and student developers and thus the scaling of the course.

CS1 provides a unique opportunity for community building. Desirable learning experiences in first-year courses and their scaling needs give rise to interesting pedagogic and software design challenges. CS programs continue to attract unprecedented talent who rightfully expect to become engaged in meaningful experiences from the first day of the program. The opportunity described in this report consists of harnessing this talent to build a sustainable community of learners. Undergraduate teaching support can be continuously recruited from the most recent pool of learners, which provides new learners with authentic "More Knowledgeable Others". Student software developers can be recruited from the same pool, which unleashes a perpetual stream of talent, motivated by their own user experience and eager to further improve and extend the system that their seniors have built.

We hope these dynamics encourage CS1 educators to engage their recent students in the teaching of incoming first-year students instead of relying on graduate students and in the development of bespoke learning tools instead of or in addition to off-the-shelf learning management systems and integrated development environments. Both activities can lead to meaningful learning experiences that enrich the first and second university years, before more advanced courses, immersive internships, and capstone projects dominate the students' university life.

In our case, the development of bespoke learning tools was facilitated by the choice of JavaScript, which was designed for building web-based systems, but languages used in CS1 today such as Python, C, Java, Racket, and OCaml are also supported by vast repositories of open-source tools and environments that enable the student-led development of learning and course management tools. We recommend a sublanguages approach, not just for its well-known pedagogic advantages, but also because it places the student-led development of programming language tools within reach of first- and second-year student projects. Apart from CS1, other courses might provide similar opportunities, including courses on data structures and algorithms, data science, and discrete mathematics.

Commercial providers of LMSs currently do not seem to prioritize the extensibility of their products for the needs of programming-related courses. We described the development of a bespoke LMS for a programming course as a drastic remedy. We pointed out that the process of developing such a system from scratch provides learning and community-building opportunities. To educators who cannot or do not wish to start from scratch, the permissive licence of Source Academy provides a starting point for their own LMS or for contributing useful components to benefit a growing community.

ACKNOWLEDGEMENTS

We acknowledge valuable comments by Gerald Jay Sussman during the second author's sabbatical at MIT in early 2022, and suggestions by Daniel Jackson and Mara Kirdani-Ryan on drafts of this paper.

REFERENCES

[1] Harold Abelson and Gerald Jay Sussman. 1996. *Structure and Interpretation of Computer Programs* (2nd ed.). MIT Press, Cambridge, MA. with Julie Sussman.

[2] Harold Abelson and Gerald Jay Sussman. 2022. SICP—Comparison edition. https://sicp.sourceacademy.org JavaScript adaptation by Martin Henz and Tobias Wrigstad with Julie Sussman.

[3] Harold Abelson and Gerald Jay Sussman. 2022. *Structure and Interpretation of Computer Programs, JavaScript edition.* MIT Press, Cambridge, MA. adapted to JavaScript by Martin Henz and Tobias Wrigstad with Julie Sussman.

[4] Boyd Anderson, Martin Henz, Kok-Lim Low, and Daryl Tan. 2021. Shrinking JavaScript for CS1. In *Proceedings of the 2021 ACM SIG-PLAN SPLASH-E Symposium (SPLASH-E '21), October 20, 2021, Chicago, IL.* ACM, Chicago, IL, 87–96. https://doi.org/10.1145/3484272.3484970

[5] Boyd Anderson, Martin Henz, and Hao-Wei Tee. 2021. Ruggedizing CS1 Robotics: Tools and Approaches for Online Teaching. In *Proceedings of the 2021 ACM SIG-PLAN SPLASH-E Symposium (SPLASH-E '21).* ACM, New York, NY, 82–86. https://doi.org/10.1145/3484272.3484969

[6] Kaian Cai, Martin Henz, Kok-Lim Low, Xing Yu Ng, Jing Ren Soh, Kyn-Han Tang, and Kar Wi Toh. 2023. Visualizing Environments of Modern Scripting Languages. submitted, under review.

[7] Chris McCord. 2018. Phoenix Framework. https://phoenixframework.org

[8] Elixir Core Team. 2019. Elixir. https://elixir-lang.org

[9] estree. 2021. estree. https://github.com/estree/estree.

[10] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2018. *How to Design Programs: An Introduction to Programming and Computing* (2nd ed.). MIT Press, Cambridge, MA.

[11] Peter Henderson. 1982. Functional Geometry. In *Conference Record of the 1982 ACM Symposium on Lisp and Functional Programming.* ACM, New York, 179–187. https://doi.org/10.1145/800068.802148

[12] Martin Henz, Shang-Hui Koh, and Samyukta Sounderraman. 2021. Teachable Moments in Functional Audio Processing. In *Proceedings of the 2021 ACM SIG-PLAN SPLASH-E Symposium (SPLASH-E '21).* ACM, Chicago, IL, 65–70. https://doi.org/10.1145/3484272.3484967

[13] Martin Henz, Thomas Tan, Zachary Chua, Peter Jung, Yee-Jian Tan, Xinyi Zhang, and Jingjing Zhao. 2021. A Stepper for a Functional JavaScript Sublanguage. In *Proceedings of the 2021 ACM SIG-PLAN SPLASH-E Symposium (SPLASH-E '21).* ACM, Chicago, IL, 71–81. https://doi.org/10.1145/3484272.3484968

[14] Richard C. Holt and David B. Wortman. 1974. A sequence of structured subsets of PL/I. *ACM SIGCSE Bulletin, Proceedings of the fourth SIGCSE Technical Symposium on Computer Science Education* 6, 1 (January 1974), 129–132.

[15] M. Homer, T. Jones, J. Noble, K.B. Bruce, and A.P. Black. 2014. Graceful Dialects. In *ECOOP 2014—Object-Oriented Programming (Lecture Notes in Computer Science, Vol. 8586)*, Jones R. (Ed.). Springer, Berlin, Heidelberg, 131–156.

[16] LEGO System A/S. 2021. LEGO MINDSTORMS. https://www.lego.com/en-us/themes/mindstorms.

[17] Meta Platforms, Inc. 2022. React—A JavaScript library for building user interfaces. https://reactjs.org

[18] Mourya Reddy Narasareddy Gari, Gursimran Singh Walia, and Alex David Radermacher. 2018. Gamification in Computer Science Education: a Systematic Literature Review Paper. In *Proceeedings of the 2018 ASEE Annual Conference & Exposition.* American Society for Engineering Education, Salt Lake City, UT, 13 pages. https://doi.org/10.18260/1-2--30554

[19] Source Academy. 2021. js-slang. https://github.com/source-academy/js-slang.

[20] Source Academy. 2022. https://github.com/source-academy.

[21] Source Academy. 2022. *Specification of JavaScript sublanguages for SICP JS.* Source Academy Specifications. Source Academy, https://docs.sourceacademy.org.

[22] Students and staff of CS1101S at the National University of Singapore. 2022. Modules for Source Academy. https://github.com/source-academy/modules.

[23] Students and staff of CS1101S at the National University of Singapore. 2022. Sources for SICP JS. https://github.com/source-academy/sicp.

[24] Benedikt Wisniewski, Klaus Zierer, and John Hattie. 2020. The Power of Feedback Revisited: A Meta-Analysis of Educational Feedback Research. *Frontiers in Psychology* 10 (1 2020), 14 pages. https://doi.org/10.3389/fpsyg.2019.03087