A MODULE SUPPORTING THE COLLABORATIVE DESIGN AND BUILD ACTIVITY

G. Brieven, A. Labrahimi Kasdaoui, B. Donnet

Uliege (BELGIUM)

Abstract

The Collaborative Design and Build (CDB) activity addresses key STEM education challenges: developing 21st century skills, scaling feedback for large classes, and enabling hybrid learning. In CDB sessions, teams of students form problem-solving chains where each team builds upon previous work. This approach suits any STEM course requiring multi-step problem solving.

Until this semester, CDB sessions were conducted using physical worksheets. We now use a digital module that manages the flow of step-by-step solutions and feedback between teams, while also supporting real-time collaboration on shared solutions within each team. In this module, instructors can configure problem statements, solution steps, criteria for feedback, group and team size, and timing of each step. When a session begins, problems are distributed among teams, solutions advance between teams with structured feedback periods, and the activity concludes with validation and retrospective analysis.

We used this module for the first time in our CS2 course where students tackled recursive problems through three steps: mathematical formulation, function specification, and implementation.

At the end of the session, we collected student perception data, mainly to compare digital versus paper-based approaches. Student feedback favored the digital CDB module (86% found its interface clear, 80% reported smoother workflow, and 83% noted improved collaboration) compared to paper-based approaches. The coding step was identified as the most suitable step for digitalization (90%). The primary challenges were time constraints and team interdependence, suggesting the need for flexible timing features and highlighting the inherent collaborative nature of the activity.

Keywords: Collaborative Problem solving, Assembly-line Learning, Hybrid learning, 21st Century Skills.

1 INTRODUCTION

In our Introduction and Complement to Programming courses (CS1 and CS2), it is crucial to prepare students to solve complex problems. The format of their final answer should be a piece of code. Upstream the development of the solution, students are taught to process it at higher levels of abstraction via specific modeling steps, promoting so a top-down approach ([1]).

In practice, students tend to skip these modeling steps and straightly code via trial and error. That may work for simple problems. But as soon as students face more complex ones, they end up with less robust solutions and longer code.

To overcome this, we motivate this top-down approach by connecting it to students' future professional life and embedding it in a Collaborative activity called CDB (standing for Collaborative Design and Build).

The Collaborative Design and Build (CDB) activity addresses three major challenges faced by STEM instructors: promoting 21st century skills (communication, collaboration, critical thinking, and problem-solving) ([2]), achieving scalability coupled with rapid feedback (supporting up to 500 students), and enabling hybrid learning through online accessibility.

In a CDB session, teams of students belong to a chain made up of problem-solving steps interspersed with transition phases. Only the first team in the chain can access the problem statement, while subsequent teams must build upon the work of previous teams to advance the solution.

This activity is suitable for any STEM course that requires multiple steps to solve problems. While previously limited to in-person classroom settings (using paper-based methods), the CDB activity is now a module that integrates with existing Learning Management Systems through LTI (Learning Tool Interoperability).

2 HOW THE COLLABORATIVE DESIGN & BUILD (CDB) ACTIVITY WORKS?

The CDB activity consists of two phases: Design and Build. This approach encourages students to reflect on a solution before developing it, to limit solution patching afterwards.

As shown in Figure 1, groups of students are divided into *T* teams, with each team working on problems in parallel through *T* given steps. The Design phase typically involves analysis and modeling, while the building phase focuses on implementation.

In our CS2 course specifically, students are progressing through three sequential steps (*T=3*). They should first provide a recursive definition of the problem, then specify the functions to implement, and finally write the actual code. Each team handles one step before passing their work to the next team.

Transition periods allow teams to provide feedback on previous work using checklists ([3], [4]), helping to maintain quality throughout the process. At the same time, they also receive feedback from their peer and should refresh their own solution step ([5]).

This structure mirrors professional environments where large projects require effective team collaboration.

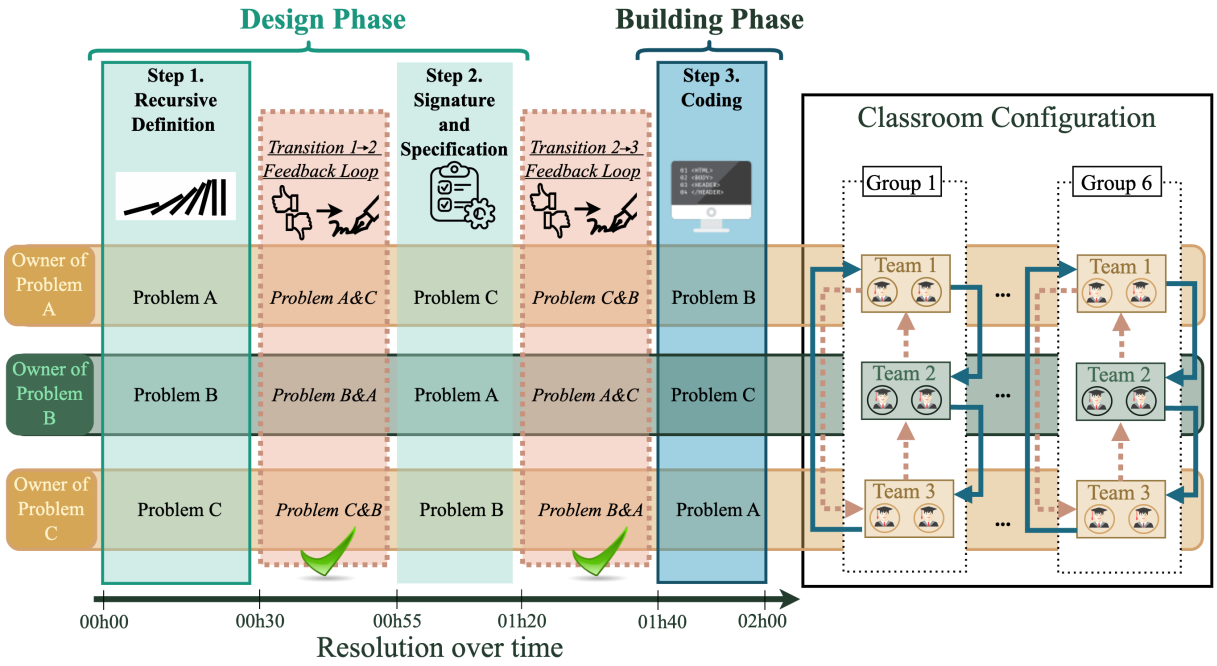

Figure 1. CDB Organization, instantiated in our CS2 course, with three problem-solving steps.

3 FUNCTIONALITIES OF THE CDB MODULE

The CDB Module provides distinct functionalities tailored for two different types of users: the *instructor* and the *student*. Functionalities are summarized in Figure 2.

3.1 Instructor View

The CDB module allows instructors to define: (i) the problem statements (i.e., uploading a PDF), (ii) the solution steps (e.g., problem decomposition, coding (in Computer Science)), (iii) acceptance criteria for each step (forming the basis for transition phases), (iv) the number of groups and team size, and (v) the duration of both problem-solving steps, and transitions from one step to another.

The instructor functionalities are primarily accessible through the *Administration Panel*, organized in three tabs: *Libraries, Sessions Preparation* (see Figure 3) and *Courses*. Next, instructors can also monitor on-going sessions via the *Running Sessions Monitoring* page.

In the current version, two types of libraries are proposed: one for predefined answers and another one for checklists. Before an instructor defines a session (in the *Session Preparation* tab), they should make

sure to have predefined the answers and checklists they need in their problem-solving flow. To do so, they should either create a new checklist/predefined answer or copy and modify an existing one. LaTeX format is also supported. Checklist and predefined answers can also be previewed or deleted.

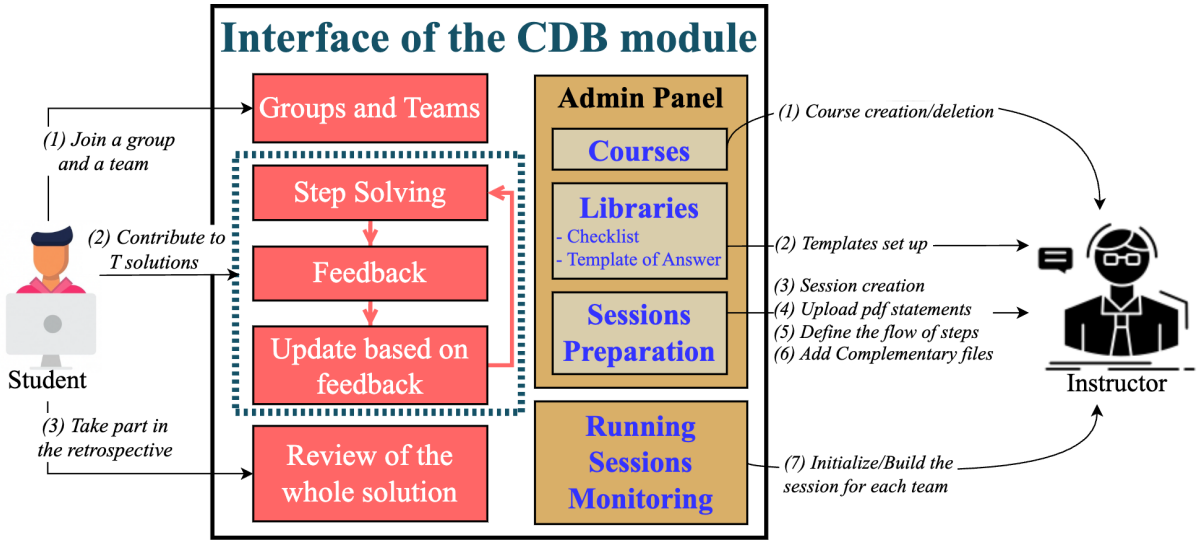

Figure 2. Overview of the functionalities of the CDB module.

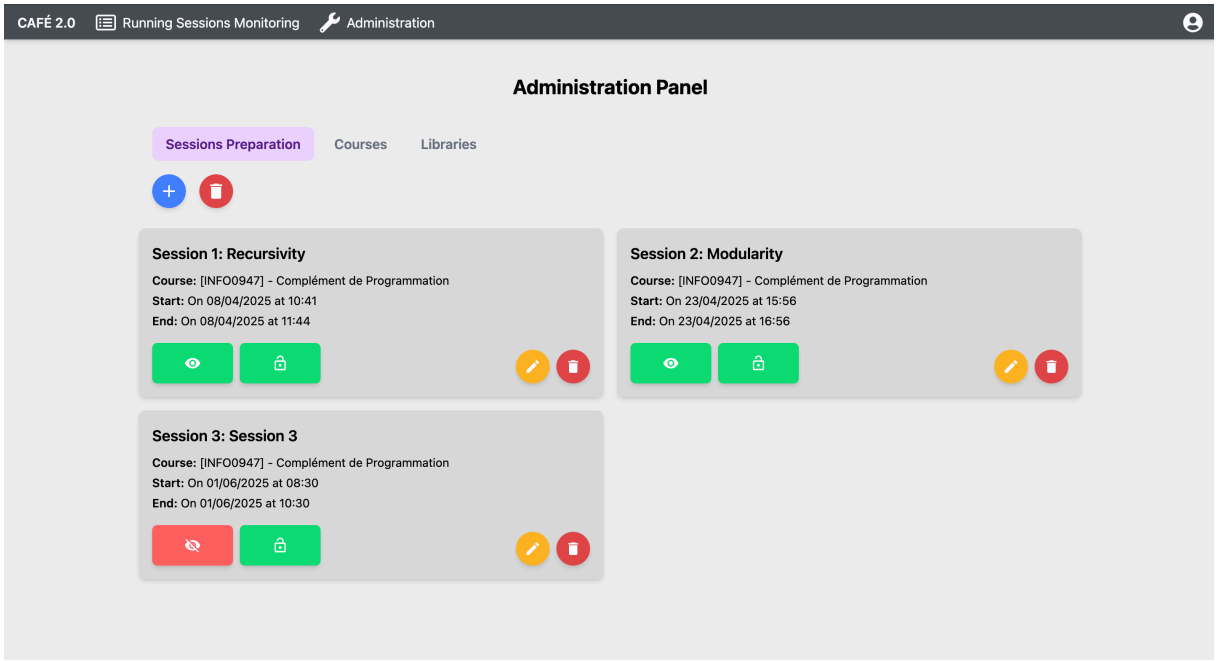

Figure 3. Administration Panel - Sessions Preparation tab.

In the *Sessions Preparation* tab, instructors have an overview of all existing sessions and can create new ones. To create a new session, instructors should first specify its main settings (title, course, number of groups, teams and team members, start and end time, description and access). Once a session has been created, its content can be specified (see Figure 4). Instructors can update the main parameters at any time before starting the session. They also must upload *T* problem statements and define *T* problem-solving steps in the *Flow of Steps* page (Figure 5). In the *Flow of Steps* page, instructors should also specify the Feedback and Correction sub steps, following each problem-solving sub step (except the last one for which it's optional). The instructor also indicates the amount of time

allocated to each sub step. Ultimately, the instructor can launch a session by clicking the Play button on the main page (accessible through the *Running Sessions Monitoring* tab, next to *Administration*).

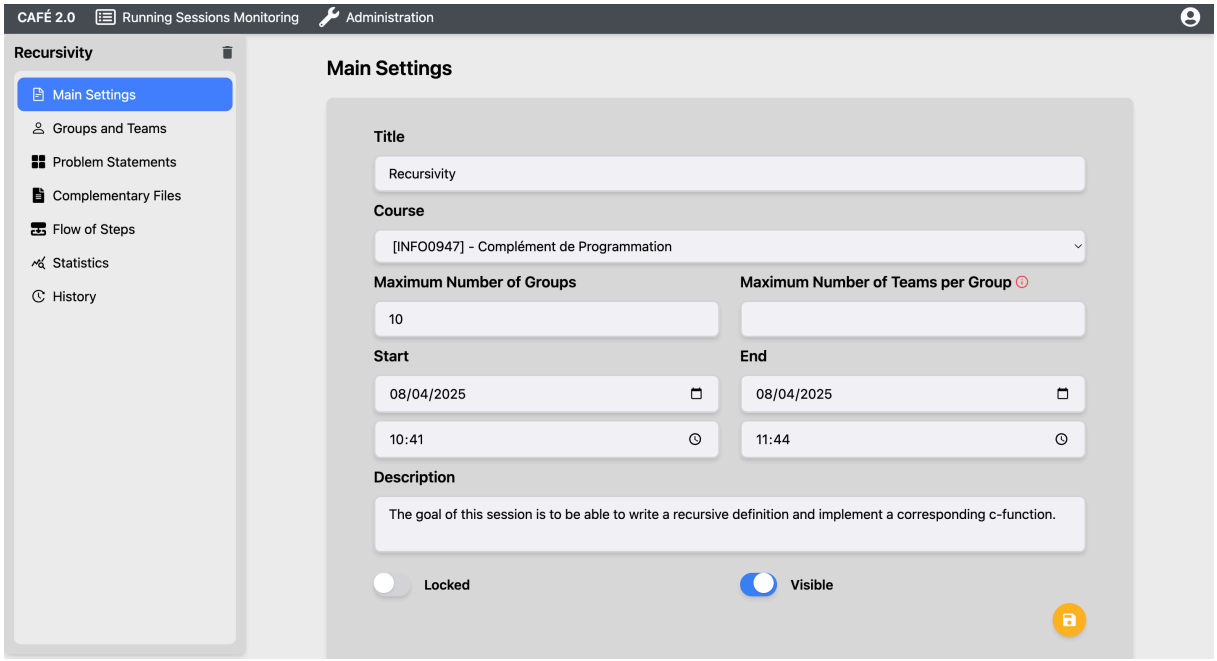

Figure 4. Main Settings Page to define a CDB session.

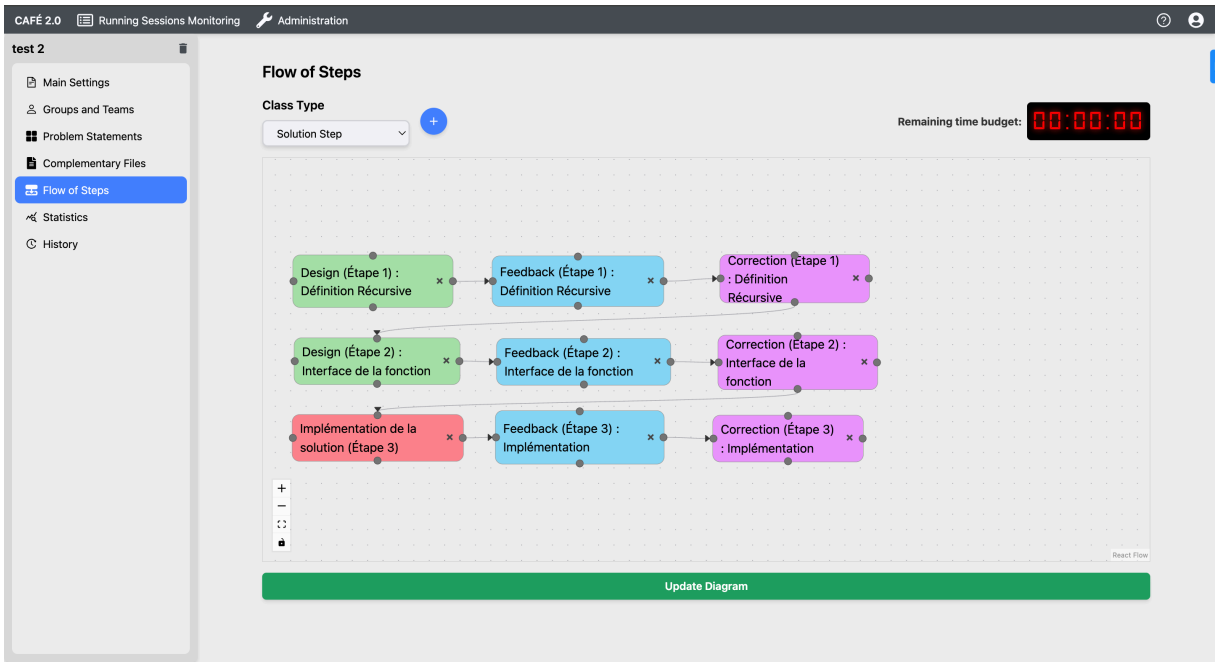

Figure 5. Flow of steps, where each step is composed of a solution part, feedback on it and the elaboration of a new version of the solution part, based on the feedback.

3.2 Student View

After instructors have configured a CDB session, students can join a group via the *Join a Team* page, and, inside the group, join a team. Once a group is complete, distinct problem statements are distributed to each team in a group for parallel solving. As illustrated in Figure 1, if *T* steps are required to solve the

problems, then *T* teams stand in a group and *T* problem statements initiate *T* problem-solving chains in each group. During the first step, the teams solve the step directly based on the problem statement (see Figure 6).

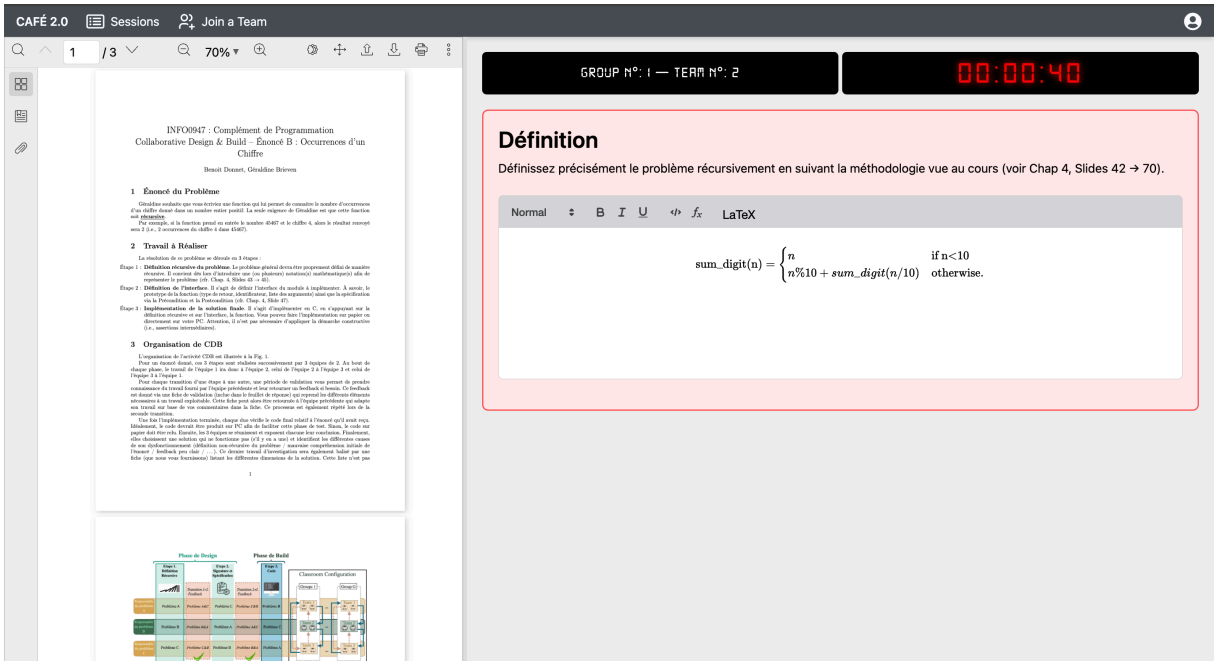

Figure 6. Students solving the first step, based on the problem statement.

When the time allocated for a step expires, step solutions advance to the next team. This team reviews it based on predetermined acceptance criteria (see Figure 7). In Figure 7, the right panel displays the checklist, automatically populated from a library template. All team members can edit the grid in real time, selecting checkboxes and entering comments against each criterion. The left panel shows the submission under review (from the previous team), while also providing access to earlier rotations' submissions for additional context.

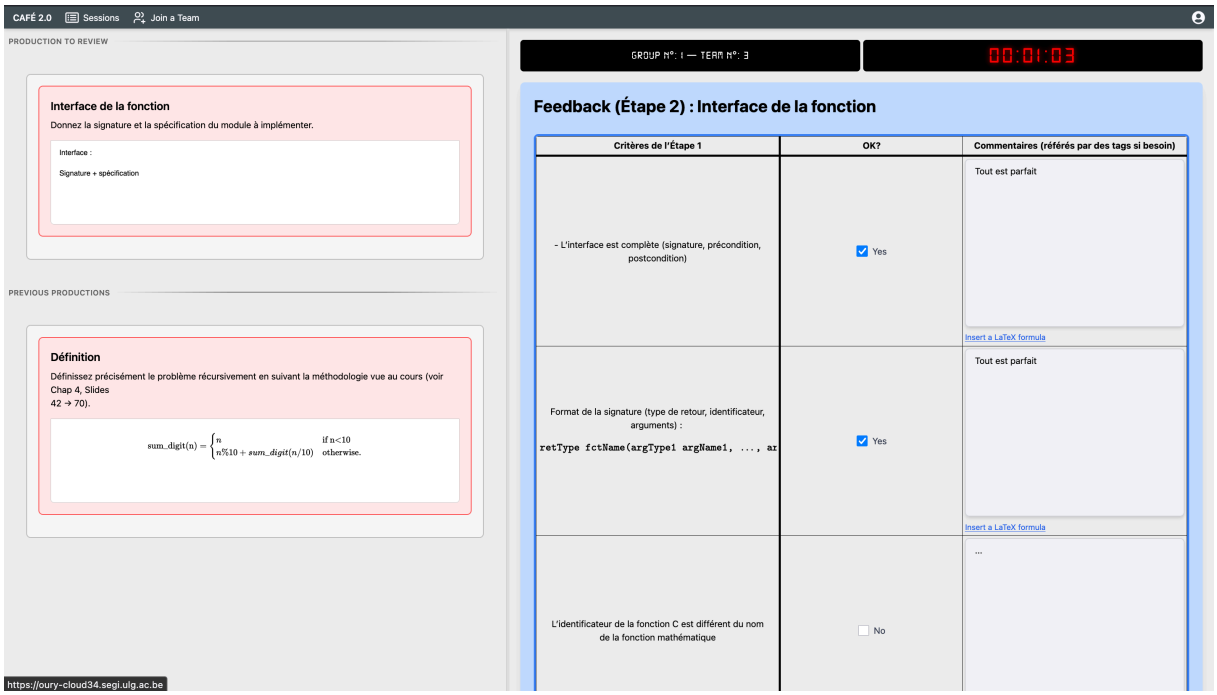

Figure 7. Students review previous team's work and fill in a checklist listing acceptance criteria.

Once feedback time is completed, step solutions return to the original team (see Figure 8). Their window is organized similarly to the feedback view. The right panel loads the submission they originally produced, allowing them to incorporate received feedback directly into their work. The left panel displays the corresponding feedback, ensuring they can see reviewer comments while making revisions. The goal of this transition phase is to maintain continuity in the problem-solving chain.

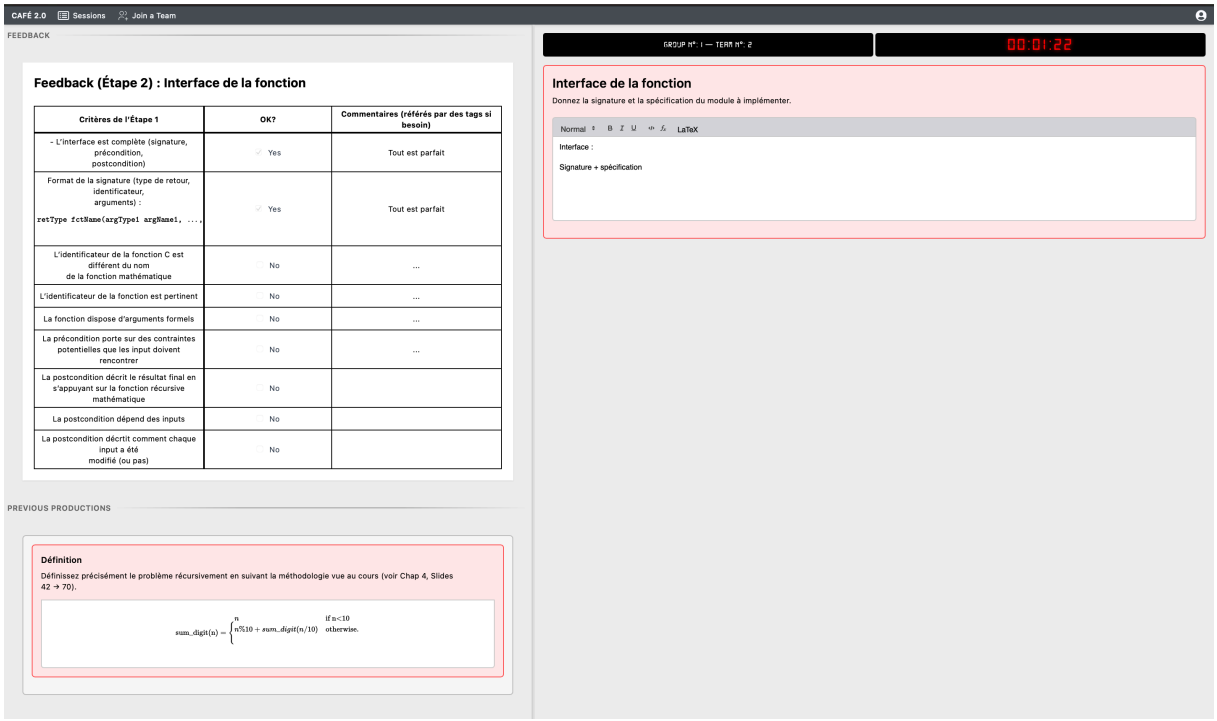

Figure 8. Students improve their step solution, based on the feedback the next team has provided.

After correction, team's work moves to the next team and the team progress the current solution by solving the next step, based on the previous one(s). The corresponding interface is illustrated in Figure 9.

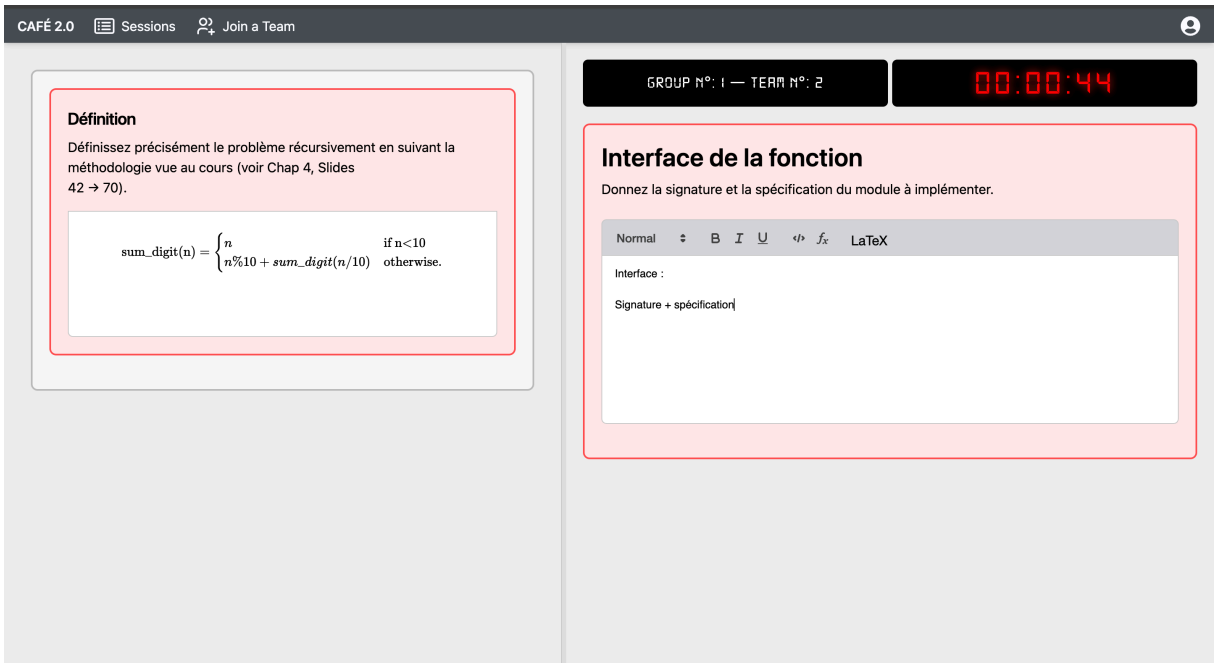

Figure 9. Students progress the solution by solving a given step, based on previous one(s) already being achieved by the previous team(s).

Finally, in the last step of our session, students translate their solutions into executable code (see Figure 10). The left panel continues to provide contextual information being the submissions from previous teams for that problem, guiding ongoing development. The right panel hosts a real-time collaborative code editor where students work together to create and modify files. Instructors pre-upload supporting resources, some marked read-only, alongside editable starter code. Beneath the editor, an integrated terminal allows immediate compilation and execution, with output and error messages displayed live. This shared workspace makes it easier for students to write, test, and improve their code.

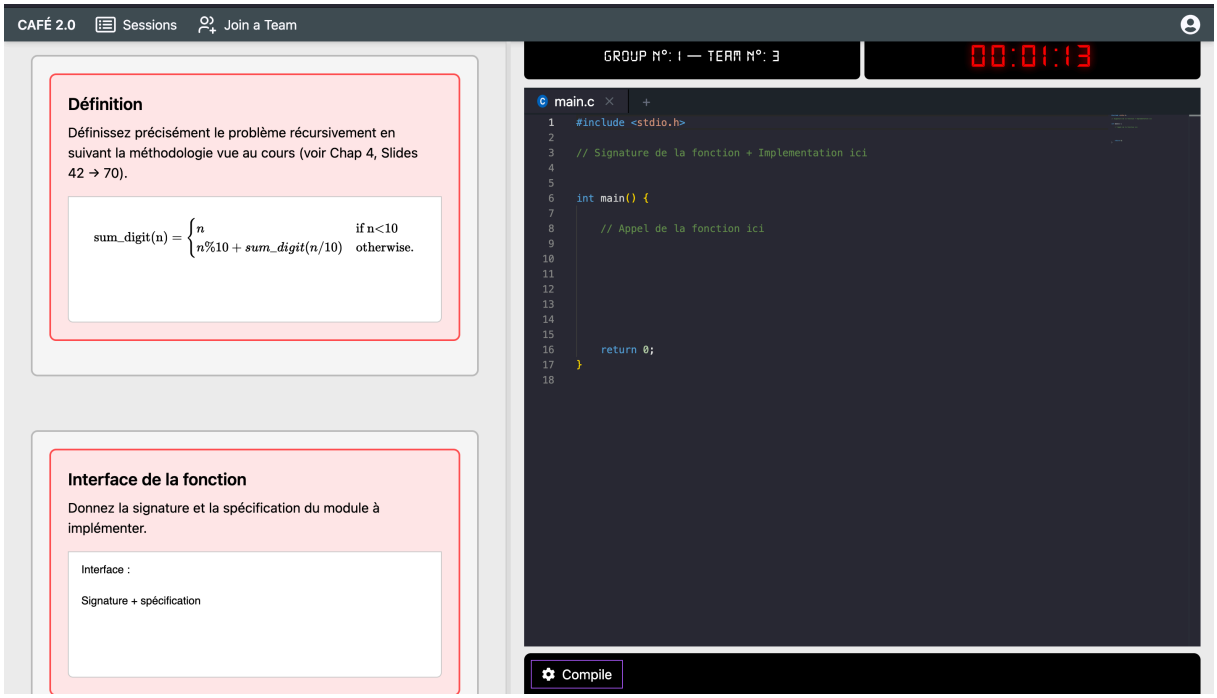

Figure 10. Students implement their solution, based on the design provided by the previous teams.

Upon completing the final step, solutions return to their initial team for validation against expected outcomes, expressed in the initial statement. Results are classified as correct, incomplete, or incorrect. Finally, a CDB session concludes with an inter-team meeting where representatives share their solution outcomes and perform a retrospective analysis (i.e., what went well and what went wrong).

4 METHODOLOGY

Prior to the implementation of the CDB module, the CDB activity was conducted entirely on paper, with students solving problems and handling peer feedback using physical worksheets. For this semester, we implemented the CDB module in our Complement to Programming course (CS2). Students tackled recursive problems through three structured steps: (i) mathematical recursive formulation, (ii) formal C-function specification, and (iii) C function implementation. The activity engaged 35 students organized into 6 groups, each containing 3 teams of 2 students (with one exception having a single student).

To assess our collaborative module's effectiveness, we conducted an anonymous post-session survey, which received responses from 30 students. All of them consented to their answers being used in this study. The survey comprised 12 questions using both open-ended formats and 5-point Likert scales. Seven questions specifically evaluated the benefits of running the activity through a digital module versus traditional paper methods. In this paper, we present findings related to this comparison and identify students' primary challenges in completing their tasks. Section 6 further discusses functionality gaps observed during this first experiment.

5 RESULTS

Overall, 86% of students rated the module's interface as clear. When comparing digital versus paper-based approaches, only 3.5% of students preferred solving problems on paper. Notably, 80% found the digital workflow smoother, and 83% reported easier collaboration with teammates thanks to shared documents with updates in real-time. However, a somewhat lower percentage (63%) preferred the CDB module over paper for supporting their solution reasoning process. This opinion is aligned with results from previous work, stating that digital formats increase the cognitive load in some cases ([6]).

When asked which solution step was most suitable for digitalization, 90% identified the development (coding) step, likely because they could immediately test their solution. Conversely, 55% struggled most with the mathematical definition step, likely due to discomfort with LaTeX expressions. We also observed the need to implement an automatic saving feature, as some students lost their work during transitions between steps due to elapsed time limits.

Regarding the question "What was the greatest difficulty during the activity?", the most common response concerned time constraints, with many students unable to complete tasks within the timeframes. This suggests implementing a feature allowing instructors to extend time for specific steps based on student time requests. Beyond timing issues, many students also noted the challenge of interdependence on teammates' work, which, while difficult, remains fundamental to the CDB activity's purpose.

6 FURTHER STEPS

As detailed in the previous section, a relevant feature would be offering a more flexible timing for each step, to limit bad quality step solution, due to a lack of time. Two students explicitly proposed this new feature. Additionally, to reduce students’ discomfort with relying on other teams, we could better balance group composition by distributing higher-performing students across different groups. To support this, a new feature could allow instructors to pre-assign students to groups and teams based on recent assignment grades and other factors ([7]). Currently, students are free to choose their own groups and teams.

To go further, now we have seen that the CDB module was running smoothly in an in-person setting, a next step could be organizing it in a hybrid setting, with some students participating online.

7 CONCLUSIONS

The Collaborative Design and Build (CDB) module represents a significant advancement in collaborative learning for STEM education. By transitioning from paper-based to digital formats, we have successfully addressed key challenges in promoting 21st century skills, providing scalable feedback mechanisms, and enabling greater accessibility through technology integration. Student feedback clearly demonstrates the advantages of our digital approach, with a strong preference for the module over traditional paper methods. The enhanced real-time collaboration capability and the immediate testing functionality in the coding step proved especially valuable to students. The challenges identified - primarily time constraints and the complexities of interdependent teamwork - provide clear directions for future improvements. Implementing flexible timing features and refining mathematical input methods will enhance the user experience. Additionally, having validated the module's effectiveness in an in-person setting, we are now positioned to extend its application to hybrid learning environments, further increasing its accessibility and impact.

We encourage educators to implement the CDB activity in their own STEM classrooms. Our module is designed to be accessible and adaptable across various disciplines, making it straightforward to organize collaborative problem-solving sessions that engage students in methodical approaches while developing essential teamwork skills. By sharing this tool with the broader educational community, we hope to contribute to more effective and engaging collaborative learning experiences.

ACKNOWLEDGEMENTS

This work is supported by the CyberExcellence project funded by the Walloon Region, under number 2110186.

REFERENCES

[1] G. Brieven and B. Donnet. "Practicing Abstraction through a Top-Down Problem-Solving Framework in a CS1 Course*", 10th International Conference on Higher Education Advances (HEAd'24)*, 2024. doi:10.4995/HEAd24.2024.17110.

[2] G. Brieven, M. Moraes, D. Pawelczak, S. Vasilache, and B. Donnet. "Integrating Soft Skills Training Into your Course Through a Collaborative Activity.", *ACM Technical Symposium on Computer Science Education (SIGCSE)*. ACM, 2025. doi:10.1145/3641554.3701877

[3] S. Bharuthram and M. Patel M, "Co-Constructing a Rubrick Checklist with First Year University Students: A Self-Assessment Tool*", Journal of Applied Language Studies (Apples)*, vol. 11, no. 4, pp. 35–55, 2017. 10.17011/apples/urn.201708073430.

[4] D. Alt and L. Naamati-Schneider and D. J. N. Weishut, "Competency-Based Learning and Formative Assessment Feedback as Precursors of College Students' Soft Skills Acquisition", *Studies in Higher Education*, vol. 48, no. 12, 1901-1917., 2023. 10.1080/03075079.2023.2217203.

[5] G. Brieven, L. Leduc, and B. Donnet, "How Students Manage Peer Feedback through a Collaborative Activity in a CS1 Course", *9th International Conference on Higher Education Advances (HEAd),* 2023. 10.4995/ HEAd23.2023.16142.

[6] P. R. Whipp, J. Pengelley, A. Malpique, "A testing load: a review of cognitive load in computer and paper-based learning and assessment", *Technology, Pedagogy and Education*, 34. 1-17, 2024. 10.1080/1475939X.2024.2367517.

[7] Chih-Ming Chen & Chi-Hsiung Kuo, "An optimized group formation scheme to promote collaborative problem-based learning.", *Computers & Education,* 2019. 10.1016/j.compedu.2019.01.011.