Codeschool in a Box: A Low-barrier Approach to Packaging Programming Curricula

Yoshi Malaise[a], Evan Cole[b] and Beat Signer[c]
Web & Information Systems Engineering Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
{*ymalaise, evan.cole, bsigner*}*@vub.be*

Keywords: Curriculum Packaging, Programming Education, Exercise Generation, Smart Learning

Abstract: The tech industry is a fast-growing field, with many companies facing issues in finding skilled workers to fill their open vacancies. At the same time, many people have limited access to the quality education necessary to enter this job market. To address this issue, various small and often volunteer-run non-profit organisations have emerged to up-skill capable learners. However, these organisations face tight constraints and many challenges while trying to design and deliver high-quality education to their learners. In this position paper, we discuss some of these challenges and present a preliminary version of a curriculum packager addressing some of these issues. Our proposed solution, inspired by first-hand experience in these organisations as well as computing education research (CER), is based on a combination of micromaterials, study lenses and a companion mobile application. While our solution is designed for the specific context of small organisations providing vocational ICT training, it can also be applied to the broader domain of learning environments facing similar constraints.

1 Introduction

The tech industry is a fast-growing field and according to Voka, the Flemish Network of Enterprises, Belgium is facing a severe shortage of technically schooled people (Voka, 2019). Belgium is hardly the only country facing these issues since according to Statista, the worldwide full-time employment in the ICT sector is projected to reach 62 million in 2023.[4] At the same time, many countries are dealing with large groups of socially vulnerable people with limited access to the job market. For example, the Belgium federal agency for the reception of asylum seekers (Fedasil) accommodated a total of 31 808 asylum seekers[5] in January 2023. These conditions gave rise to multiple initiatives all over the world, trying to provide up-skilling opportunities to people with limited access to the labour market by training them in specific highly demanded skills in their local ICT industries. A few examples of such organisations are HackYourFuture (Denmark, The Netherlands and Belgium), MigraCode or Girls Who Code.

Unfortunately, due to the shortage of ICT-skilled people, there also exists a shortage of career and technical education teachers capable of teaching these courses (Devier, 2019). This implies that these organisations are often run by people with little to no technical experience, and courses are taught on a volunteer basis by tech professionals who do not necessarily have any pedagogical experience. Furthermore, these organisations cannot even fall back on open university courses as research indicates that universities are often not addressing all the industry's specific needs (Akdur, 2022) and mainly offering long-form programmes, making the material a less than ideal fit for our target audience.

In order for a curriculum packaging solution to best serve learners, it is vital that the management is not distracted from operations and the progress of their learners, and that volunteer mentors can focus on what they know best; working one-to-one with learners and preparing examples of well-written code. We set out to create a solution that makes it easy for volunteers to create materials following educational best practices. It should further be easy for educational directors to remix and reuse existing online materials. In this paper, we outline a low-barrier approach to packaging programming curricula based on our own experience and inspired by computing education research (CER).

[a] https://orcid.org/0000-0002-3228-6790
[b] https://orcid.org/0000-0001-8190-0446
[c] https://orcid.org/0000-0001-9916-0837

[4] https://www.statista.com/statistics/1126677/it-employment-worldwide/

[5] https://www.fedasil.be/en/statistics

2 Context

In order to highlight the novelty and importance of the work presented in this paper, it is important to introduce the specific context in which we want the proposed tools to operate, how this context might differ from the traditional university-level courses or k-12 education that are typically studied by computing education research, and how existing freely available resources do not address the needs of vocational ICT training. Let us start by formulating some key characteristics of the stakeholders in this context.

The *learners* in our target audience are typically people who do not have access to full-time continued education. They are typically following these upskilling classes in combination with a full-time job and are often also taking care of a family. Therefore, the presented content should be flexible for asynchronous consumption when it fits within their schedule and they cannot always have the guidance of a teacher in the room. Having content accessible in multiple forms that can be accessed in different contexts is a major benefit for these learners.

In our context, *content creators* are well-meaning software developers who want to contribute to education as volunteers by making content in the domain of their expertise available to the previously-mentioned learners. These volunteers are a very valuable resource as their expertise carries a lot of value to the learners, but they are frequently not trained as educators. Often, they also do not have experience with what makes good educational content and what the typical progression of students looks like. Therefore, we want to make sure that they can spend their time doing what they do best, creating code samples for the material they want to cover. We then need to include a system to automatically create exercises of varying difficulty levels about this content so students can progress naturally. More invested volunteers can also be a major help if we guide them in the creation of what good online resources look like. Using this guidance, they can contribute by developing a small application instead of directly writing content.

The *curriculum designers* are typically members of organisations that want to combine these open education resources into a holistic curriculum covering enough content to prepare learners to take an entry-level position in a specific role, such as a web developer dealing with in-demand frameworks. The curriculum designers often have limited pedagogical or technical expertise, and given the fact that they are likely part of a small non-profit organisation with limited funding and popular technologies evolve quickly, they normally do not have the luxury to spend a year in designing a brand new curriculum. Given these constraints, it is important for curriculum designers to find adequate resources—often created by the volunteer developers mentioned earlier—that can be reused and repurposed to fit their needs.

Even though these constraints have a major impact on what kind of material might work and what will not (precluding many theoretically ideal approaches), we feel that it is valuable to specifically target this challenging context, given that these learners are not reached by traditional university education, and they stand to gain the most by switching to a stable job in the information technology sector. Note that everything that we present can also form an added value for regular education.

3 System Components

In the following, we introduce the key components necessary to properly construct a curriculum for our target audience. These components consist of a set of independent online resources focusing on a narrow learning objective, while providing immediate feedback in order for students to practise and improve even when no teacher is nearby, a new way to explore existing source code in order that students can learn from pedagogically sound content generated from samples provided by content creators, and a mobile companion application that allows students to practise some exercises in a gamified environment while they are on the go and do not have access to a desktop computer or laptop.

3.1 Micromaterials

In the field of education, we have units of learning materials called *learning objects* (LO), with each learning object having well-defined goals on what they are trying to teach. When authors of these learning objects take the necessary precautions to ensure that they can be openly reused by other educators, these learning objects might be referred to as *open education resources* (OER). The 2019 UNESCO definition describes OER as *"teaching, learning and research materials that make use of appropriate tools, such as open licensing, to permit their free reuse, continuous improvement and repurposing by others for educational purposes"*.[6] It is clear from this definition that it is not enough to simply be freely available to others to be classified as an OER, but it should be realistic to incorporate these resources into existing

[6] https://opencontent.org/definition/

larger curricula as educators see fit. The computing education research field has already made great strides towards not only developing some of these resources, but also making them available to a larger community. For example, every year authors can submit nifty assignments that they used in their lectures to the nifty assignments track of the Technical Symposium of the ACM Special Interest Group on Computer Science Education (SIGCSE) and present them to their peers. It is important to note that during the submission process authors have to provide additional metadata about an assignment, such as the intended target audience, additional context or its strengths as well as weaknesses. This additional metadata is of vital importance for educators who do not simply want to use existing open education resources as they are, but possibly repurpose them for their own situations. After the conference, the artefacts are uploaded to the nifty assignments page of Stanford University[7] where they can be found with all the necessary metadata.

There is a subset of open education resources referred to as *micromaterials*, a term coined by Adam Leskis.[8] In order for an open education resource to classify as a micromaterial, there are additional constraints that need to be satisfied. A micromaterial should be directly usable by the learner and the micromaterial should be able to provide automated feedback so the learner can improve without additional oversight by an educator. This way of working is a perfect match with our target audience of learners who might be practising at home after a workday. Although it is not a strict requirement to be considered a micromaterial, many existing micromaterials contain automatically generated content and are mobile friendly. This provides the user with a virtually endless amount of content they can work through until they grab the concept that they are trying to learn.

In order to further illustrate the idea of a micromaterial, we provide three examples of micromaterials that have been developed to be used by our students.

- **King's Scroll**
  In this browser-based game students are shown a randomly generated piece of JavaScript code that modifies four boolean variables (helmet, sword, shield and cape) (Malaise and Signer, 2023). They have to interpret the piece of JavaScript code and select the hero that satisfies the constraints of these variables at the end of execution of the code. The micromaterial is used to help students practise the control flow of JavaScript programs.

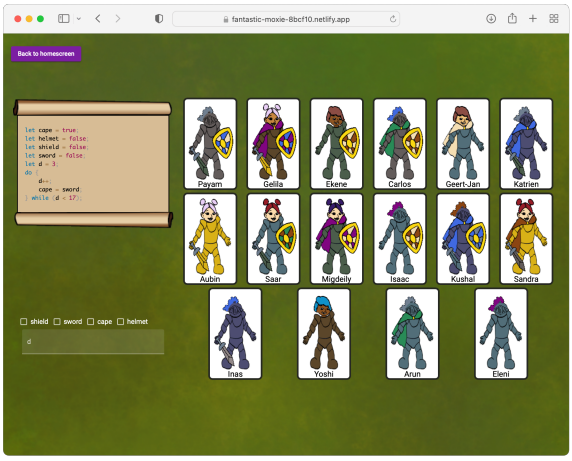

Figure 1: Screenshot of King's Scroll micromaterial

- **SQL Study Buddy**
  This browser-based application uses sql.js[9] (an SQLite version transpiled to WebAssembly) and allows students to write queries to be executed in their browser. This helps to lower the barrier as nothing needs to be installed locally and students cannot mess up their database given that it is recreated for every exercise. Students are asked to write SQL queries to perform certain operations in the online code editor. Every time a student executes their query, a set of assertions are run. By looking at which assertions pass and which fail, students can keep improving their queries until all checks pass.
- **HTML Study Buddy**
  HTML Study Buddy is a browser-based application to practise the basics of HTML markup. Students are presented a screen that is divided into three parts. In the right part of the screen, they see a webpage that they have to recreate. Further, in the left part of the screen, they are presented jigsaw puzzle pieces based on Google's Blockly library. Those jigsaw pieces contain specific HTML markup. The available pieces are determined by the elements used on the page the students have to recreate, to prevent them from being overwhelmed. Students can snap the pieces together in order to construct the webpage. The webpage resulting from the markup on the assembled jigsaw pieces is updated in the centre of the screen as soon as a student drags one of the pieces.

3.2 Study Lenses

As discussed earlier in Section 2, there exists an ample number of online sample projects for nearly every

[7] http://nifty.stanford.edu

[8] https://micromaterialsblog.wordpress.com

[9] https://sql.js.org

given topic in programming. Programmers like to write blog posts about new technologies that they think are exciting which can also be beneficial for them to attract new employers. The main issue however is that these posts are typically not written by pedagogical experts and the sample code does oftentimes not have actionable goals for students to learn from in a step-by-step manner. This led to the idea of *study lenses*.[10]

The concept of study lenses is straightforward; we want to empower students to be able to learn from *any online code the learner encounters*, but still do so in a pedagogically sound way. In order to achieve this goal, raw source code is used as input for different *lenses* and each lens is used to generate materials for a specific learning objective (e.g. reading or tracing). A wide variety of views are possible and one of the more basic examples can highlight the code's syntax and overlay a canvas to enable students to annotate and highlight parts of the code. A different lens might take the code file, parse it and present the code as a flowchart to help students focus on the control flow, or the execution of the program could be visualised in tools such as PythonTutor[11] to help student focus on what goes on in the program's memory during execution. Some micromaterials can also be packaged as lenses. For example, we have lenses that ask students questions about their code or lenses that introduce errors in the code that a student has to fix. The study lenses environment builds on a plugin-based architecture where educators and developers can introduce new lenses on demand.

When designing new lenses, there are a couple of important constraints to take into consideration. First, it is important that lenses work on *plain code*. Any code should run fine in the target environment and content creators should not need to change the way they write the examples to fit the study lenses environment. Second, lenses should be designed with a *"peel away"* principle in mind, where early lenses can provide a lot of additional context on top of the code, but the more advanced lenses should do so less and less, in turn revealing a general-purpose development environment that matches realistic industry setups. This way we can satisfy the expertise reversal (Kalyuga et al., 2009) and the skill transfer (Chiaburu and Marinova, 2005) principles. Finally, lenses should not create any platform lock. Content created to be used with the lenses should not rely on any custom syntax and the content should still be usable (to some extent) in a different environment. This is why there is a strong push towards using open web standards that can be reused everywhere in addition to regular code files for all aspects of curriculum packaging.

The current implementation of study lenses is available open source and as a Node.js package that can be installed globally on a student's system. Students can run the `study` command from any place in their file system to explore the current directory in the study lenses environment. There also exists a study lenses version that can be embedded into packaged curricula as described later in Section 4.

3.3 Companion App

The need for mobile learning environments has been discussed in the past (Göksu and Atici, 2013) and its importance has also been highlighted to us during conversations we had with students of different upskilling courses. Many students emphasise the usefulness of having access to some material on the go when they happen to have some time available to engage with content during free time, such as when they are taking public transportation. Therefore, we implemented an initial prototype of a mobile companion application; not to replace the study lenses environment or the online resource, but as an additional way of consuming content. In order to make the application fully portable, we opted for a smartphone application that stores its content offline and thus can be used even without any cellular connection.

The mobile app has two main functionalities. First it can be used to view (short) presentations about important topics as a refresher, which might be useful to do right before a class or before students start working on their laptops when getting home. Second the app also includes small exercises that are generated from code files. The app further includes some gamification elements such as push notifications to remind students to get their daily practise, and daily goals such as completing 10 exercises every day. After completing a certain number of exercises, badges can be unlocked and shown in the application. Further, exercises are presented according to the Leitner box system where exercises that have never been completed have the highest chance of being selected, and the more often an exercise has been successfully completed, the less frequently it is shown. In the following we list the currently supported types of exercises.

- **Parsons Problem**
  Parsons problems have been used for a long time in computing education research (Du et al., 2020). When given a Parsons problem, students receive all the lines of a code snippet in a shuffled form. By dragging and dropping the lines, students need to recreate the original code snippet. Whenever a

[10] https://github.com/colevandersWands/study-lenses/

[11] https://pythontutor.com

line is placed at the right position, a green checkmark is shown on that line. This way, users incrementally work on the problem until they have solved the exercise.

- **Comment Slots Exercise**
  In a comment slots exercise, a user is presented a snippet of code, but all the comments have been replaced with combo boxes. It is up to the student to interpret the code and match every comment with the corresponding line of code.
- **Parameter Chooser**
  In a parameter chooser exercise, users are presented with a function that takes at least two parameters. However, all occurrences of the parameters in the body will be replaced by combo boxes. By reading and understanding the surrounding code, it is up to the learner to infer which of the parameters is used.
- **Code Snippet to/from Flowchart**
  Code Snippet to/from flowchart exercises present the user with an original code snippet and three flowcharts. One of the flowcharts is the correct translation of the code snippet, the other two are based on slightly altered versions of the code snippet. It is up to the user to determine which of the three flowcharts matches the original snippet. An inverted version of this exercise is also possible where users receive one flowchart and three pieces of JavaScript code.
- **Multiple Choice Questions**
  The multiple choice question exercises have been inspired by *"questions about learners' code (QLCs)"* (Lehtinen et al., 2021). In these exercises, the application analyses a code snippet shown to a student and asks them questions about the code. Examples of questions include *"What is the name of the function?"*, *"Is function X asynchronous?"* or *"Which of the following variables are declared in loop initialisers?"*.

4 Curriculum Packager

There is a strong need for many organisations to set up new courses as well as to update existing ones. These courses could be both long-term formation such as 'Introduction to Web Development' over a period of up to a year, or an advanced follow-up course such as 'Reactive Programming' with a duration of two months only. It is essential for organisations to be able to set up these smaller curricula in order to accommodate for the specific professional needs in their geographical area.

In recent years, research on how to best package a curriculum has been conducted by the Curriculum Materials Working Group.[12] However, their research mainly targets long-form traditional university courses making use of existing *learning managements systems* (LMS). Similar research lead to open standards such as SCORM,[13] that allow moving material between LMS's. However, such a setup is not a perfect fit for our intended use case of ad-hoc volunteers, working together to come up with a set of open learning materials. The management of an LMS requires a substantial effort and the time learners spend on learning to work with the LMS is time they could have spent on learning about the concrete content. Therefore, we strive for our learning environment to be as close to the industry tools that coaches and learners will be using throughout their careers. The preliminary version of the packager combines the components introduced in the previous section into a holistic self-contained online curriculum that can be deployed to any cloud hosting service. Note that our packager currently supports JavaScript-based curricula, but we plan to add other programming languages.

In our solution, the source of a curriculum is made up from a set of folders, with each folder representing one learning object (e.g. LO_1) as illustrated in Figure 2. Such a learning object can consist of any type of files, but the following files are given a special meaning during the packaging process:

- **`Readme.md`**
  The `readme.md` file is used to describe general information about the learning object in markdown format. Typically, it is recommended for the description to list the main goals of the learning object and to refer to relevant online resources such as videos or blog posts. We also strongly encourage any learning object to link to relevant micro-materials using the provided markdown functionality. Since the system uses traditional readme files, the learning objects can easily be used outside of our environment and content developers do not need to learn any custom syntax.
- **`curriculum.json`**
  The optional `curriculum.json` file contains metadata for the packager. The `name` property can be used to set how the name of the learning object should appear in the generated curriculum. The `ignoreList` property tells the exercise generator which files to ignore when generating exercises. Further, the `requires` property can be used to indicate which learning objects should be com-

[12] https://cssplice-cm.github.io
[13] https://scorm.com

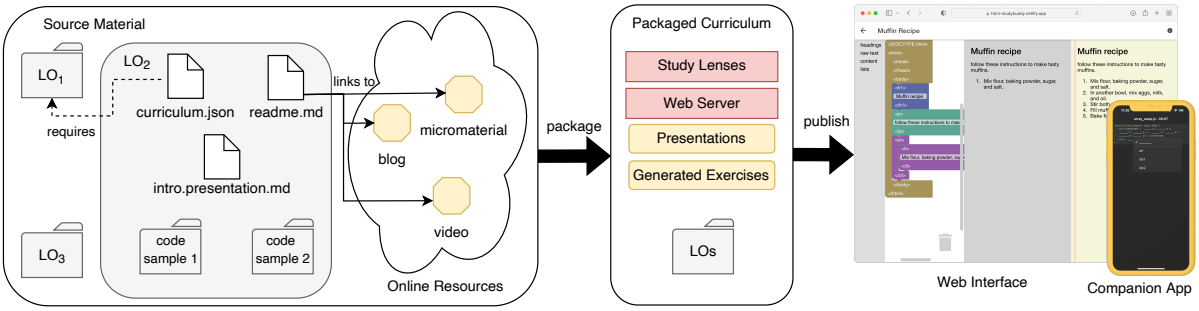

Figure 2: Curriculum packager overview

pleted before the user can partake in this learning object. The idea for this metadata was inspired by Harper-Lite,[14] a similar system that strives to make lesson (learning object) discovery easier by having authors include a `harper.yaml` file in the root directory of each lesson.

- **`study.json`**
  Optionally, a `study.json` file can be placed in any folder with code samples (e.g. `code sample 1`) to configure the default options and lenses to be used by the study lenses environment.

- **`*.presentation.md`**
  Files with a filename `*.presentation.md` will be interpreted as presentation files by the curriculum packager. The content of the file should contain valid markdown that will be turned into a presentation by the revealjs[15] JavaScript library.

- **`*.js`**
  All JavaScript files detected in the folder will be parsed and analysed in order to generate exercises based on the code. In future versions of the curriculum packager, we also plan to support other programming languages.

As we can see from the previous description, there is no need for curriculum authors to become specialised in custom tools. Their main work consists of coming up with realistically looking examples of the code in action. The markdown files can easily be reused in different contexts regardless of what learning environment is being used. This way we can avoid a vendor lock-in to our solution. Since all learning objects are simple folders that can be checked-in to GitHub, we satisfy all requirements that open learning objects should have according to the Curriculum Materials Working Group, namely that we should be able to create, fix, revise, reuse, share, find, track, pull, push, evaluate and credit open learning materials.

[14] https://third-bit.com/ideas/harper/
[15] https://revealjs.com

Once the curriculum designers have collected the learning objects they wish to include in their curriculum, they can run our curriculum packager desktop application and select all the relevant folders. The packer will then run through these folders and generate a prepackaged curriculum. The generated curriculum will be a Node.js project using an Express web server to host all its content. This project can then easily be deployed online or be made available on GitHub for students to run it locally. When accessing content from the web server, users are presented with all the learning objects in card form. On these cards, users can see the content of the `readme.md` file. Further, there are buttons to access the content of the learning object via the included study lenses server or to watch any of the presentations on the online slide presentation screen. At the top of the page there is a QR code that can be scanned by the companion application to download the curriculum's material (i.e. the mobile exercises and the presentation) for offline use.

In a future version of the packager, we are planning to add support for automatically generating learning spaces on GitHub based on workflows refined over several years of running ICT courses on GitHub. These would provide a low maintenance alternative to traditional learning management systems. Organisations willing to run a new class would be able to specify a list of students, and the system would automatically generate a GitHub repository where students can find an outline of their syllabus, indicate their progress on the Kanban board and post issues if they get stuck, while using *the same tools and workflows they will use in their professional career*.

5 Challenges and Future Work

Let us now discuss some of the main challenges that we expect to face when further developing the curriculum packager, as well as some interesting research directions that we plan to address in the near future.

5.1 The Reusability Paradox

Some criticism has arisen from researchers about the dangers of taking learning concepts out of their original context. The claim is that the context itself forms part of the material to learn, and that material that is so context agnostic that it can be used anywhere actually does not provide any real value anywhere (South and Monson, 2000). This problem has been named the re-usability paradox and represents one of the main challenges when building a curriculum out of existing components and designing the reusable components. This is something that one always has to keep in mind. However, research has shown that there definitely is still value in the creation of reusable components in education as long as effort is done to find the right balance between de-contextualisation and usefulness (Wiley et al., 2004). This means that we should not go overboard by reducing the size of the individual topics and that it is okay that some individual objects still cover more than strictly one topic (Bart et al., 2019). A rule of thumb proposed by Wiley is that we should ask ourselves *"Can you imagine wanting to teach some portion of this topic without teaching the other parts?"*. If the answer is no, all the subtopics belong to one learning object. We believe there is still room to construct a set of guidelines on the scoping of learning objects that can be shared with other researchers interested in designing reusable learning objects.

5.2 Automatic Exercise Generation

A future direction we wish to further explore are other ways to automatically generate exercises based on codebases. Ideally, these exercises should cover all aspects of the PRIMM (Predict-Run-Investigate-Modify-Make) principle (Sentance et al., 2019) to ensure that students can improve step by step over the given codebase. In our current implementation, all exercise generation is done by parsing the JavaScript files using Acorn[16] and analysing the abstract syntax tree. We are convinced that this technique can be used to generate even more interesting exercises besides the ones we are generating today. Apart from this method, we are also looking into other ways to generate exercises based on machine learning. Interesting research towards automatic exercise generation based on the code-trained Natural Language Model Codex presented by OpenAI (Chen et al., 2021) has recently been presented at ICER (Sarsa et al., 2022). This is definitely another promising direction that we are planning to further investigate.

[16]https://github.com/acornjs/acorn

5.3 More Ways to Look at Code

We will keep advancing the study lenses platform and search for interesting and useful new lenses that could benefit our learners. We feel like the concept of using existing files as a basis and providing students different ways of interacting with and analysing these files, is an interesting concept to further explore. The versatility of the study lenses and the additional content they provide out of the box—as long as there is some example code given in a domain—leads to a great amount of new content for learners to take in with a small time investment needed by content creators.

In its current form, the study lenses implementation works as a web environment, which is great given the flexibility and ease-of-use this provides. We are further investigating what it could look like to have the plug-in-based study lenses concept incorporated into existing Integrated Development Environments (IDEs). This would not only allow students to benefit from the layers of interactivity and perspectives on top of the code *right where they are already working* but it would also allow the students to take this environment with them and use those tools to help them understand work related code as they graduate towards their first internships.

5.4 Knowledge and Content Modelling

In recent years, there has been a push towards the use of *knowledge graphs* in education. A knowledge graph is a formal way to describe all the topics and their relations that are covered within a field of education. The topics are represented as nodes, while the directed edges indicate which topics are prerequisites of other topics (Rizun, 2019). This representation allows for automated reasoning and might enable future intelligent tutoring systems. In past work, the knowledge graph representation has been used to find ways on how to model students' progression (Ilkou and Signer, 2020), as well as to build recommendation engines suggesting only exercises in a learner's zone of proximal development, both in traditional education (Baker et al., 2020) as well as in sports education (Malaise and Signer, 2022).

We are planning to further investigate the use of student modelling to adapt the content to the learners. This modelling will consist of two parts. First, we will model the learners' *constraints and preferences* (i.e. "What devices do they have access to?" or "Can they study for long sessions or do they typically have multiple short sessions?"). Second, we will also use *personal knowledge graphs* to model the progression of individual learners throughout the

global knowledge graph. Based on this additional information, we would like to introduce an intelligent tutoring system in order to better recommend exercises to the students. One can, for instance, imagine that the application does not simply show exercises based on the Leitner box system, but makes informed suggestions based on a student's past performance for specific individual skills. It might further be used to detect a user's knowledge gaps. In this domain, the EduCOR ontology[17] could be a good fit, as it models both the education side of things but also incorporates the mapping to labour-market skills. Recently, GraphBRAIN has been used to power an initial implementation of an intelligent tutoring system (Ferilli et al., 2022). GraphBRAIN allows users to utilise ontologies as database schema on top of a graph database. Exploring the direction of having our model conceptualised as an ontology (in combination with EduCOR) is a promising direction as it means that all generated data could easily be shared across systems with other projects in the field of computing education research.

6 Conclusion

In this position paper we presented a number of challenges and constraints faced by small organisations offering professional training and up-skilling for learners who are not served by traditional education. We introduced a preliminary version of a curriculum packager combining micromaterials, study lenses and a companion mobile app to address some of these challenges. It is important to note that most of the content designed to work in the presented context and its constraints will also be usable in general continued education. We further discussed some remaining challenges and future research directions. The proposed research is essential for reaching under-served learners and to expand the body of knowledge in computing education research.

REFERENCES

Akdur, D. (2022). Analysis of Software Engineering Skills Gap in the Industry. *ACM TOCE*.

Baker, R. et al. (2020). The Results of Implementing Zone of Proximal Development on Learning Outcomes. In *Proc. of EDM 2020*.

Bart, A. C., Hilton, M., Edmison, B., and Conrad, P. (2019). The Problem of Packaging Curricular Materials. In *Proc. of SIGCSE 2019*.

[17] https://tibonto.github.io/educor/

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv.

Chiaburu, D. S. and Marinova, S. V. (2005). What Predicts Skill Transfer? An Exploratory Study of Goal Orientation, Training Self-Efficacy and Organizational Supports. *International Journal of Training and Development*, 9(2).

Devier, B. H. (2019). Teacher Shortage and Alternative Licensure Solutions for Technical Educators. *The Journal of Technology Studies*, 45(2).

Du, Y., Luxton-Reilly, A., and Denny, P. (2020). A Review of Research on Parsons Problems. In *Proc. of ACE 2020*.

Ferilli, S., Redavid, D., Di Pierro, D., and Loop, L. (2022). An Ontology-driven Architecture for Intelligent Tutoring Systems with an Application to Learning Object Recommendation. *IJCISIM*, 14.

Göksu, İ. and Atici, B. (2013). Need for Mobile Learning: Technologies and Opportunities. *Procedia - Social and Behavioral Sciences*, 103.

Ilkou, E. and Signer, B. (2020). A Technology-enhanced Smart Learning Environment Based on the Combination of Knowledge Graphs and Learning Paths. In *Proc. of CSEDU 2020*.

Kalyuga, S., Ayres, P., Chandler, P., and Sweller, J. (2009). The Expertise Reversal Effect. *Educational Psychologist*, 38(1).

Lehtinen, T., Santos, A. L., and Sorva, J. (2021). Let's Ask Students About Their Programs, Automatically. In *Proc. of ICPC 2021*.

Malaise, Y. and Signer, B. (2022). Personalised Learning Environments Based on Knowledge Graphs and the Zone of Proximal Development. In *Proc. of CSEDU 2022*.

Malaise, Y. and Signer, B. (2023). King's Scroll: An Educational Game to Practise Code Prediction. In *Proc. of SIGCSE 2023*.

Rizun, M. (2019). Knowledge Graph Application in Education: A Literature Review. *Folia Oeconomica*, 3(342).

Sarsa, S., Denny, P., Hellas, A., and Leinonen, J. (2022). Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proc. of ICER 2022*.

Sentance, S., Waite, J., and Kallia, M. (2019). Teaching Computer Programming With PRIMM: A Sociocultural Perspective. *Computer Science Education*, 29(2-3).

South, J. B. and Monson, D. W. (2000). A University-wide System for Creating, Capturing, and Delivering Learning Objects. *The Instructional Use of Learning Objects*.

Voka (2019). Voka: "The Shortage of Technical Profiles is Threatening to Become the Achilles' Heel of Innovation in Flanders". Stanley Milton.

Wiley, D., Waters, S., Dawson, D., Lambert, B., Barclay, M., Wade, D., and Nelson, L. (2004). Overcoming the Limitations of Learning Objects. *JEMH*, 13(4).