Bridging the Computer Science Teacher Shortage with a Digital Learning Platform

L. M. van der Lubbe[a], S. P van Borkulo[b], P. B. J. Boon, W. P. G. van Velthoven and J. T. Jeuring[c]
Freudenthal Institute, Utrecht University, P.O. Box 85170, 3508 AD, Utrecht, The Netherlands

Keywords: Computer Science, Secundary Education, Co-Teach Informatica, Learning Platform, Student Model.

Abstract: Although computer science is important in the many aspects of the world around us, computer science education is insufficient, partly due to a shortage of qualified computer science teachers. Co-Teach Informatica offers a temporary solution to high schools in the Netherlands to overcome this shortage. With online learning materials, local remote support desks, and guest teachers, it offers a computer science course following the curriculum guidelines. This paper presents the design of an online learning platform specifically designed for the Co-Teach Informatica program. The design has two important pillars: independent learning and progress tracing. Both are important to ensure that students can follow their computer science course without a qualified computer science teacher in their classroom. Finally, we discuss the design of the student progress tracing component using a student modelling approach and the requirements of the different user groups.

1 INTRODUCTION

Computer science is an important subject because of the influence of the discipline on the world around us. The number of jobs in software development keeps on growing. For example, the demand for software developers in the manufacturing sector is now exceeding the demand for production workers (Code.org, 2017a). Research from Code.org found that "Computing occupations are the largest category of new wages in the United States – ahead of management, healthcare, finance, engineering, sales, or any other category" (Code.org, 2016). Although the skills involved in computer science are important to prepare for future working and everyday life, computer science education is insufficient.

In many countries, there is a teacher shortage for computer science, both in higher and secondary education (Shein, 2019). Moreover, in some cases, a non-expert teacher (for example a teacher certified in mathematics) is teaching computer science in a secondary school (Goode, 2007). There are different causes of this shortage, among which is the difference in salary for teaching compared to industry jobs for computer scientists (Shein, 2019). Moreover, the majority of computer science students do not consider teaching as their prior career choice when graduating (Yeni et al., 2020), as found in a research among university students from the US. Computer science is not the only subject with a teacher shortage, globally, other science subjects, such as mathematics, suffer from the same problem (McVey and Trinidad, 2019). However, the number of qualified computer science teachers who graduate is much lower compared to mathematics and science teachers (Code.org, 2017b). Different strategies can be applied to attract more teachers, among which are financial incentives such as higher salaries and grants, and more opportunities for teacher licensing for example with an emergency license until the full license is achieved (McVey and Trinidad, 2019).

Because schools often only offer a few computer science courses, there is generally only a single teacher. Thus, computer science teachers do not belong to a larger section within their school. This can create practical challenges and makes collaborations between teachers difficult (Goode, 2007).

Computer science is an elective course in most countries, and many schools are not offering it (Yeni et al., 2020). This means that not all students will be able to attend a computer science course at their school or university (Shein, 2019). The content of computer science curricula in high schools greatly varies per country and can also differ between schools. It varies from a focus on basic digital skills

[a] https://orcid.org/0000-0003-1678-5159
[b] https://orcid.org/0000-0001-6668-5282
[c] https://orcid.org/0000-0001-5645-7681

van der Lubbe, L., van Borkulo, S., Boon, P., van Velthoven, W. and Jeuring, J.
Bridging the Computer Science Teacher Shortage with a Digital Learning Platform.
DOI: 10.5220/0011971900003470
In *Proceedings of the 15th International Conference on Computer Supported Education (CSEDU 2023) - Volume 1*, pages 289-296
ISBN: 978-989-758-641-5; ISSN: 2184-5026

with a vocational emphasis to learning foundational computational thinking skills and design principles in a problem-solving context (Goode, 2007). Therefore, students can have different skills at the end of their high school education, preparing them for future jobs or studies to different extents. In the Netherlands, Computer Science is also an elective course and few schools are offering it. There is a national curriculum for the upper levels of high schools, however schools still have freedom of how to translate this to their lessons. Thus, although some general baseline knowledge is required for all students, there are still differences in the knowledge and competencies of the students completing a high school course on Computer Science in the Netherlands.

Reducing the teacher shortage takes time, and until the situation has improved temporary solutions are needed. To bring computer science to high schools without a qualified computer science teacher, Co-Teach Informatica[1] offers a remotely organized computer science program following the curriculum for Dutch schools. Partly, their program is delivered via online materials. This paper presents the design of the online learning platform designed specifically for the Co-Teach Informatica program in the Netherlands. The design has two important pillars: independent learning and progress tracing. A teacher at a school participating in the Co-Teach Informatica program has no or limited content knowledge, but rather coaches students and manages the practicalities of the course. Therefore, the platform should facilitate the learning of students in the best way possible. Moreover, a teacher cannot support students in their progress due to this lack of content knowledge. Tracing the learning progress of the students with the use of learning goals can help the teachers, but also the students themselves to gain more insights into their knowledge level and potential knowledge gaps.

To ensure that the design of the platform is suitable for the target group and the program, it is designed in an iterative process and with close collaboration with different stakeholders. This paper describes the design of the platform. The remainder of this paper will be as follows: The background section describes the Dutch computer science curriculum, the organization of the Co-Teach Informatica program, and methods to trace students' progress. In the section '*Platform Design*', the design of the platform is described in more detail. In the '*Future Work*' section, we briefly look forward to the studies that will be performed to improve and evaluate the platform. Finally, in the '*Conclusions*' section, the importance of the research is stressed with a prospect of results that can be expected from future studies.

[1]https://www.co-teach.nl/

2 RELATED WORKS

2.1 Dutch Computer Science Curriculum

In the Netherlands, computer science is offered as an elective course in the higher grades of general secondary education (in Dutch: havo) and pre-university education (in Dutch: vwo) (Grgurina et al., 2018). While regular subjects in the curriculum end with a national exam in the final grade, computer science does not have such an exam. Instead, schools can create their own exams within the given guidelines. The content of the curriculum for secondary computer science education was renewed in 2019 to be more up-to-date. The curriculum consists of mandatory core domains and elective domains. There are three publishers for learning material for computer science in the Netherlands: Instruct, Informatica Actief and VO-Content. Often teachers mix materials from different publishers and other sources, including their own materials.

2.2 Co-Teach Informatica

Many vacancies for high school computer science teachers in the Netherlands are unfulfilled and this number will only increase in the coming years. Schools are therefore forced to stop offering computer science when their teacher leaves or are unable to start teaching it (Lucassen, 2022). Co-Teach Informatica aims to help those schools with a temporary solution following the official curriculum. For the mandatory domains, Co-Teach Informatica designed online learning materials that students can use independently while a school teacher is present to manage and motivate the class and do the administration for the course. This school teacher is not qualified for computer science and does not need to have computer science knowledge. To support students on the learning content, remote local support desks of teaching assistants and subject didactics experts are organized by Co-Teach Informatica. Multiple remote support desks are each dedicated to schools in a specific region, hence they are called local remote support desks. The teaching assistants also correct and grade the work of the students. For the elective domains, guest teachers who are IT professionals are invited to give a project-based module in a class. Those guest teachers receive didactical training from Co-Teach Informatica. An overview of the organization of the Co-Teach Informatica program can be found in Figure 1.

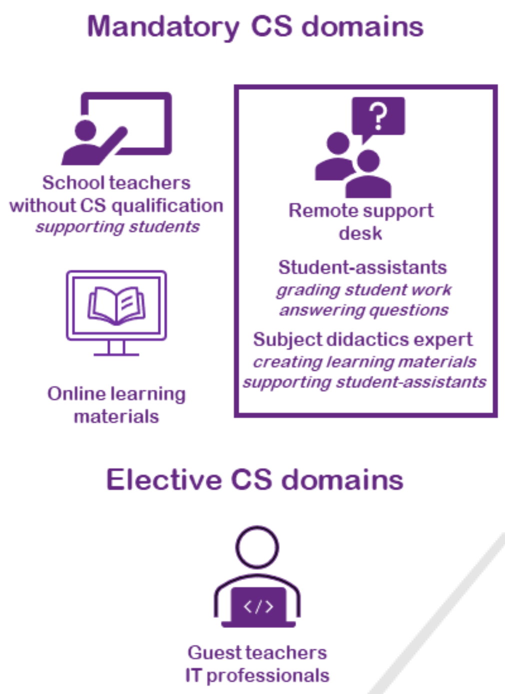


Figure 1: Overview of the organization of the Co-Teach Informatica program.

This paper presents the design of a learning platform developed for the Co-Teach Informatica program. The focus of this platform is to facilitate students, teachers and local remote support desks in their learning process and work. The platform has two important pillars: independent learning and progress tracing. These two pillars are particularly important because the teacher at the participating school is not a computer science content expert and therefore has limited content knowledge. Therefore, it is important that the independent learning of the students is supported in the best way possible. That means for example that communicating with the local remote support desk should be easy and accessible. Although the teacher present in the classroom can motivate and guide students, the teacher might not be able to oversee their learning progress as the teacher is not familiar with the content. Tracing the students' progress by modelling their knowledge based on learning goals can help to give students, teachers, and teaching assistants insights into the progress of students and classes.

2.3 Tracing Student Progress

To follow the knowledge state of a student and to predict future performance, knowledge or skill modelling can be used (Pelánek, 2017). This is the most used form of what is called student modelling. Besides a student's knowledge and skills, other aspects can be incorporated into a student model, such as student motivation, preference or affect. While those aspects are dynamic, static aspects like gender or age can also be used (Chrysafiadi and Virvou, 2013).

Student models are useful for teachers or educational content developers to gain insights in for example item difficulty or learner clusters (Pelánek, 2017). Learner clustering puts students into a cluster of similar students. The system that uses this clustering can adapt to those clusters, instead of treating all students as if they were coming from a homogenous group. For example, if student clusters are created around learning difficulties, each cluster can receive different forms of information.

Making a student model available to students offers them a tool for reflection and self-monitoring, supports discussions between students and teachers and supports self-regulated learning (Pelánek, 2017). In such cases, the model is often visualized, for example in a (directed) graph. In adaptive educational systems, student models are a way to provide personalization for individual students (Chrysafiadi and Virvou, 2013). Based on the information in the student model about the needs, preferences, and knowledge state, the system can adapt the learning path.

A student model can cover different aspects: the process of learning and forgetting, using observational data, domain modelling, and learner clustering and individualization (Pelánek, 2017). There are different approaches to modelling learning and forgetting. For this type of modelling, there is a distinction between models that are based on assumptions about learning, such as the Bayesian knowledge tracing that includes forgetting, item difficulty and individualization, and models without these assumptions about learning that instead use approaches such as (exponential) moving averages. For both types of models simple and more complex approaches exist. Observational data is often the correctness of an item, but can also be the response time, use of hints, wrong answers or the history of attempts. To model the domain, there are different ways in which the knowledge components (items such as learning material or exercises) can be organized. Knowledge components can be organized in disjoint sets, meaning that each item belongs to one learning component. However, it is also possible that multiple knowledge components apply to one item. Knowledge components can also be arranged in a hierarchy or a structure with prerequisites. Basic student models assume that all learners come from one homogeneous population. However, in reality, learners are often diverse which can be explored with learner clustering.

Intelligent tutoring systems are digital tools that can be used as a (partial) alternative to one-to-one human tutoring and can be used for computer science education. For example, intelligent tutoring systems give feedback on the syntax or semantics of code written in special environments (Crow et al., 2018). In general, intelligent tutoring systems have two important roles for which student modelling is used: a diagnostic role and a strategic role (Chrysafiadi and Virvou, 2013). The diagnostic role means that the system understands the knowledge of the student. The strategic role means that the system can plan a response to that knowledge state. In their review of intelligent tutoring systems for programming education, Crow et al. (2018) found that such systems often lack reference materials for the students and mainly focus on programming tasks and the required instructions. Reference material is often closely related to the domain model, which is an important component of a student model. User interactions with reference material can normally be used as input for the student model. Although intelligent tutoring systems for programming often lack reference material, it is still possible to use student modelling, for example for generating hints and feedback. However, the underlying knowledge domain model might not be visible to the student because it cannot be linked to explaining resources and therefore it might be more difficult to understand the hints and feedback. When reference material is available, students can seek clarification based on the feedback of the student model (Crow et al., 2018).

3 PLATFORM DESIGN

This section describes the design and the rationale behind the design of the learning platform. Important prerequisites for the newly designed platform are allowing independent learning and tracing the students' progress, while supporting the specific workflow of Co-Teach Informatica. The emphasis on student progress tracing, using a student model approach, makes this platform unique compared to other existing platforms. In the next paragraph, the rationale behind this is explained in more detail. In the following paragraphs, the main features to facilitate independent learning within the Co-Teach Informatica program are explained per user type, i.e., the learners, the support desk, and the teachers.

3.1 Student Progress Tracing Using Student Modelling

To trace the progress of the student, a student model approach is used. This student model represents the knowledge state of the student and is based on a domain model consisting of all learning goals of the learning materials. This domain model is organized on different levels. Learning goals are connected with other learning goals to represent prior knowledge. For each mandatory domain of the curriculum, the learning goals are organized by chapter. This helps to reduce the complexity of the graph and allows easy and comprehensible filtering. Connections with prior knowledge can go beyond the learning goals of the chapter or even the domain.

All learning activities in the online learning materials are linked to learning goals. For each learning goal, a so-called probability score can be calculated based on the student's achievement on activities related to the given learning goal. The probability score expresses the probability that a learner will answer the next question linked to the learning goal correctly. Every time a student completes an activity, the probability score for the linked learning goals (and their prior knowledge) changes according to the evaluation of the activity. The exact mathematics behind this will evolve during initial testing. Figure 2 shows an example of what a learning goal graph for a programming chapter could look like.

As mentioned in the background section, student models have two roles in intelligent tutoring systems: a diagnostic role and a strategic role. In our platform, the student model also has these two roles.

When our student model is used as a diagnostician it is mainly a tool for reflection, both for the learner and the teacher or the local remote support desk. Aside from being a tool that can be used to give them insight into their progress, it can also stimulate self-regulated learning. Within the learning material, there are different types of activities: exercises for practising and so-called milestone assignments. Exercises for practising are not mandatory and are often automatically corrected or manually assessed by the learner. Milestone assignments are obligatory and larger activities, often corrected by teaching assistants. Learners are free to decide whether they want to follow the learning material linearly and complete (all) the exercises for practising before starting with the milestone assignment or cherry-picking theory and exercises to prepare for the milestone assignment. The information on the learning goals can help them to make informed decisions about this.

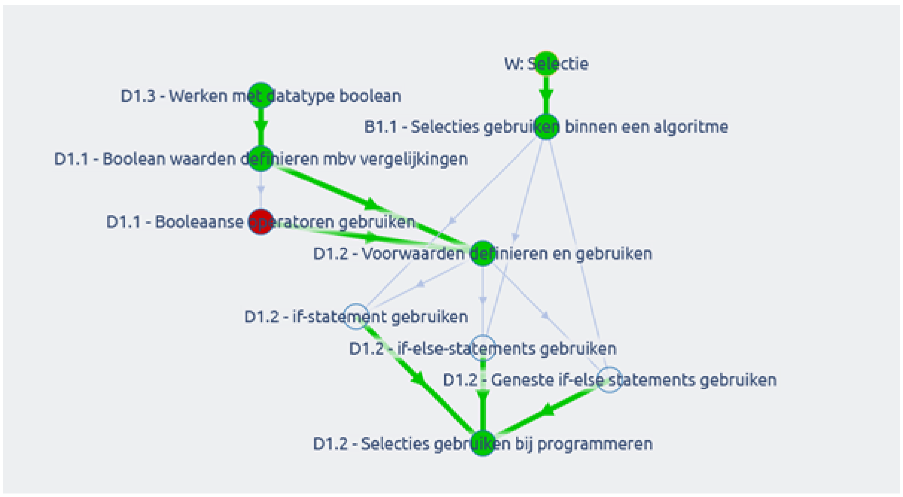

W: Selectie
D1.3 - Werken met datatype boolean
B1.1 - Selecties gebruiken binnen een algoritme
D1.1 - Boolean waarden definieren mbv vergelijkingen
D1.1 - Booleaanse operatoren gebruiken
D1.2 - Voorwaarden definieren en gebruiken
D1.2 - if-statement gebruiken
D1.2 - if-else-statements gebruiken
D1.2 - Geneste if-else statements gebruiken
D1.2 - Selecties gebruiken bij programmeren

Figure 2: Example learning goal graph for a programming chapter.

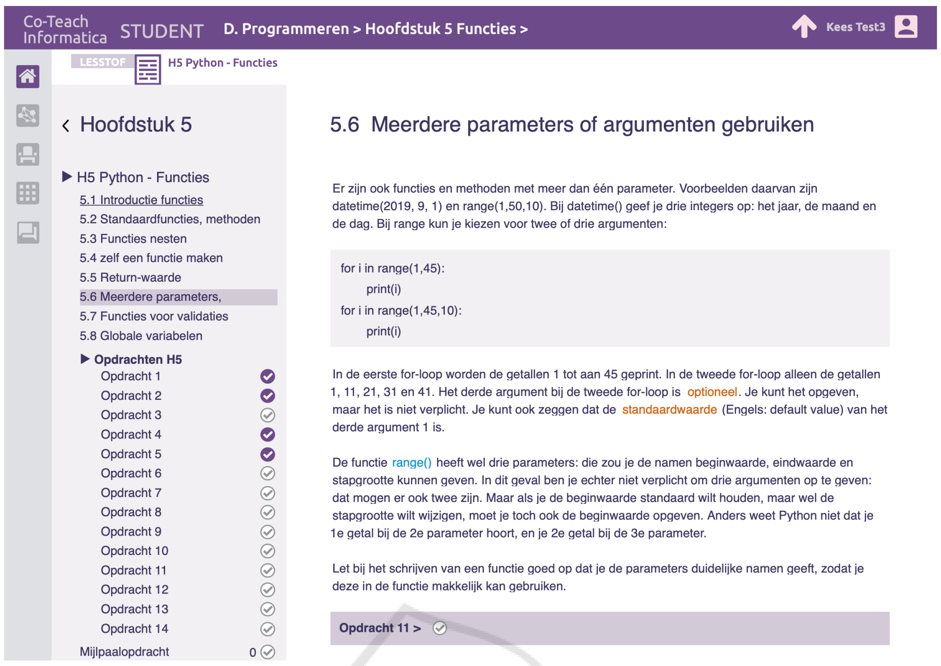
The use of our student model can also be extended towards having a strategic role. Instead of only giving the learners the freedom to customize their learning path by for example skipping exercises for practising, the system could also actively make recommendations to the learners. With these recommendations, the system can help learners to increase their probability score (and thus their knowledge) on weaker learner goals. For example, if a learner has a probability score lower than 50% for multiple learning goals for the next milestone assignment, the system can recommend exercises that cover those learning goals or prior knowledge if that is still lacking. This can help learners to prepare for milestone assignments or exams. This role has the potential to be extended to other types of recommendations, such as recommending supporting peers that can complement each other.

The prerequisite for the first application of the student model is that all the learning material can be represented in the domain model. This means that all learning activities are linked to nodes in the domain model graph. Tweaking the mathematics behind this model, such as weight factors representing the impact of an item evaluation on the probability score, can be done based on experiences in the classroom. For the second way of applying the student model, the prerequisites are more complicated. To be able to recommend suitable exercises for learners, there needs to be a significant number of exercises. There have to be multiple ways in which a learning goal can be practised, and there have to be exercises using different combinations of learning goals to be able to fill knowledge gaps. The current Co-Teach Informatica program material does not include such a variety of exercises yet.

3.2 Learners' Requirements

Learners use the system to access their learning materials, including theory, exercises for practising, milestone assignments and exams. Figure 3 shows the student view of learning material on the platform. Exercises and milestone assignments can be completed within the platform. Short answers can be given through text boxes, different closed answer types are facilitated and Jupyter Notebook is integrated to allow students to work on Python code. For some exercises, like writing a report, or when specific software is needed, exercises are solved outside of the platform and can be submitted to the platform manually. Exercises are assessed in three different ways: automatically (for example for multiple choice questions), manually by the learners (with the help of example answers), or by the local remote support desk (in the case of milestone assignments). In the last case, learners can read the feedback on their exercises in the platform and hand in improvements if necessary. Exams can be taken and graded via the platform as well.

Aside from accessing the learning materials, learners can follow their progress in different ways. All exercises for practising are marked with checkboxes that indicate whether the exercise is completed (see Figure 3). This gives learners a clear overview of their progress within a chapter but does not say anything about whether the exercise was correct or not. Because the exercises are meant for practicing and are not graded, the score is less important, and thus the focus is on completing them. For milestone assignments, which are graded, the checkbox shows more information. A score (0-100%) is shown together with a green or red checkmark indicating whether the



‹ Hoofdstuk 5

▶ H5 Python - Functies
5.1 Introductie functies
5.2 Standaardfuncties, methoden
5.3 Functies nesten
5.4 zelf een functie maken
5.5 Return-waarde
5.6 Meerdere parameters,
5.7 Functies voor validaties
5.8 Globale variabelen

▶ Opdrachten H5
Opdracht 1
Opdracht 2
Opdracht 3
Opdracht 4
Opdracht 5
Opdracht 6
Opdracht 7
Opdracht 8
Opdracht 9
Opdracht 10
Opdracht 11
Opdracht 12
Opdracht 13
Opdracht 14
Mijlpaalopdracht 0

5.6 Meerdere parameters of argumenten gebruiken

Er zijn ook functies en methoden met meer dan één parameter. Voorbeelden daarvan zijn datetime(2019, 9, 1) en range(1,50,10). Bij datetime() geef je drie integers op: het jaar, de maand en de dag. Bij range kun je kiezen voor twee of drie argumenten:

```
for i in range(1,45):
    print(i)
for i in range(1,45,10):
    print(i)
```

In de eerste for-loop worden de getallen 1 tot aan 45 geprint. In de tweede for-loop alleen de getallen 1, 11, 21, 31 en 41. Het derde argument bij de tweede for-loop is optioneel. Je kunt het opgeven, maar het is niet verplicht. Je kunt ook zeggen dat de standaardwaarde (Engels: default value) van het derde argument 1 is.

De functie range() heeft wel drie parameters: die zou je de namen beginwaarde, eindwaarde en stapgrootte kunnen geven. In dit geval ben je echter niet verplicht om drie argumenten op te geven: dat mogen er ook twee zijn. Maar als je de beginwaarde standaard wilt houden, maar wel de stapgrootte wilt wijzigen, moet je toch ook de beginwaarde opgeven. Anders weet Python niet dat je 1e getal bij de 2e parameter hoort, en je 2e getal bij de 3e parameter.

Let bij het schrijven van een functie goed op dat je de parameters duidelijke namen geeft, zodat je deze in de functie makkelijk kan gebruiken.

Opdracht 11 >

Figure 3: Screenshot of the platform from the student's view.

score is sufficient to pass or whether another attempt is needed.

In addition to this, learners can follow their progress via a learning goal graph, as shown in Figure 2. For each chapter, the part of the learning goal graph with the relevant learning goals, coloured according to their probability scores for a student, is shown interactively so that students can explore it. For every learning goal, they can read a description, find an example and link to the relevant reference material.

Lastly, the platform allows learners to communicate with the local remote support desk, the teacher, the class group, or other individual learners. An integrated chat-like message service is offered to facilitate these different ways of communication.

3.3 Local Remote Support Desks' Requirements

The local remote support desk uses the platform to prepare learning materials and make them available for different classes.

As explained earlier, some of the learners' work is corrected by the teaching assistants from the local remote support desk. The work of students can be downloaded or viewed on the platform and a rubric can be filled in to evaluate the work. This rubric is also linked to learning goals, to strengthen or weaken the probability scores of those learning goals (see Figure 4).

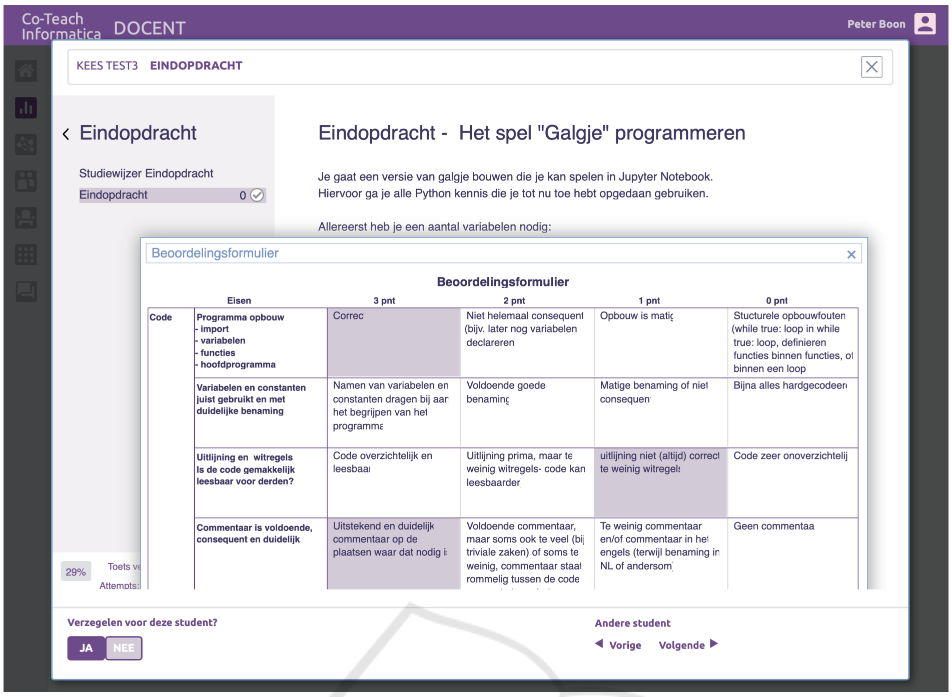
The local remote support desk can communicate with individual learners, class groups, teachers and each other via the integrated chat service.

3.4 Teachers' Requirements

Teachers have the same user type (technically) as the local remote support desk. However, teachers will not use the possibility to change learning material and grade assignments. Teachers need to edit the planning as this is class specific and can follow the progress of their students. Moreover, they can communicate with their class, individual learners or the local remote support desk.

4 CONCLUSIONS AND FUTURE WORK

The platform that is introduced in this paper will ultimately be used within the Co-Teach Informatica program. To develop the platform, intermediate evaluations will ensure an informed design. Therefore, the stakeholders (students, teachers, and local remote support desks) are involved in the design process. After initial brainstorming to get insights into the user requirements, the prototype of the system was built. This prototype will be evaluated in a pilot study, in which multiple classes will be using the platform for a short course on programming. The research with



Eindopdracht - Het spel "Galgje" programmeren

Je gaat een versie van galgje bouwen die je kan spelen in Jupyter Notebook.
Hiervoor ga je alle Python kennis die je tot nu toe hebt opgedaan gebruiken.

Allereerst heb je een aantal variabelen nodig:

Beoordelingsformulier

Beoordelingsformulier

	Eisen	3 pnt	2 pnt	1 pnt	0 pnt
Code	**Programma opbouw - import - variabelen - functies - hoofdprogramma**	Correc	Niet helemaal consequent (bijv. later nog variabelen declareren	Opbouw is matig	Stucturele opbouwfouten (while true: loop in while true: loop, definieren functies binnen functies, of binnen een loop
	Variabelen en constanten juist gebruikt en met duidelijke benaming	Namen van variabelen en constanten dragen bij aan het begrijpen van het programma	Voldoende goede benaming	Matige benaming of niet consequen	Bijna alles hardgecodeer
	Uitlijning en witregels Is de code gemakkelijk leesbaar voor derden?	Code overzichtelijk en leesbaa	Uitlijning prima, maar te weinig witregels- code kan leesbaarder	uitlijning niet (altijd) correct te weinig witregels	Code zeer onoverzichtelij
	Commentaar is voldoende, consequent en duidelijk	Uitstekend en duidelijk commentaar op de plaatsen waar dat nodig i	Voldoende commentaar, maar soms ook te veel (bij triviale zaken) of soms te weinig, commentaar staat rommelig tussen de code	Te weinig commentaar en/of commentaar in het engels (terwijl benaming in NL of andersom	Geen commentaa

Verzegelen voor deze student?

JA NEE

Andere student

◀ Vorige Volgende ▶

Figure 4: Screenshot of a rubric to evaluate the work of students.

this platform will give insights into how independent learning within a classroom setting can be facilitated. Requirements for the platform, and prerequisites that need to be met to fulfil these requirements, will become clear.

Since this is still a limited prototype, the student model will be limited to having a diagnostic role. Future work will focus on incorporating more learning material to extend the roles of the student model. Moreover, the exact embedding and visualization of the student model is an important focus point of the first pilot study. Aside from this, certain design choices, such as the way that completion of exercises for practising and milestone assignments are shown to the learners are important questions for this pilot study. Finally, the pilot study aims to improve the user experience for all involved stakeholders to best facilitate their workflow.

Aside from the user experience with the platform, the goal is also to integrate a student model. Our research will give insights into how a student model can be applied to the Dutch Computer Science curriculum. Moreover, a suitable way of visualizing this student model and the way that it is embedded in the material is also explored in this research. Those findings apply to other courses as well.

For the prototype design, the scope of the platform with respect to the learning content is still limited. The learning materials of one mandatory domain (programming) are included, which also means that the student model is only developed for that domain. In the future, the learning materials will include all mandatory domains. Each domain within the Dutch computer science curriculum will have its demands for the platform. While programming is a rather practical domain, other domains can be more theoretical, which may pose other challenges. For example, domain model graphs representing the student models might become less connected. We also expect that extending the learning material in the platform creates the need to change the initial design or extend it with new requirements. Thus the platform needs to be versatile, which makes it also broader applicable, possibly for other subjects as well.

The platform is designed to be used within the Co-Teach Informatica program, but our platform could also be used by qualified computer science teachers that want to follow their students' progress using a student model based on learning goals. And in addition, we see potential for it to be used in other contexts as well. More secondary school subjects are suffering from teacher shortages. It might be interesting to apply a similar approach using the infrastructure of our platform for those subjects to help overcome the problem of teacher shortage more broadly.

ACKNOWLEDGEMENTS

The research reported here was part of a larger project, (partly) financed by EFRO and REACT EU, project number KVW-368.

EUROPESE UNIE
Europees Fonds voor Regionale Ontwikkeling.
Mede gefinancierd in het kader van de respons van de Unie op de COVID-19-pandemie.

REFERENCES

Chrysafiadi, K. and Virvou, M. (2013). Student modeling approaches: A literature review for the last decade. *Expert Systems with Applications*, 40(11):4715–4729.

Code.org (2016). Computing occupations are now the #1 source of new wages in america.

Code.org (2017a). In manufacturing, software jobs outpace production jobs for first time.

Code.org (2017b). Universities aren't preparing enough computer science teachers.

Crow, T., Luxton-Reilly, A., and Wuensche, B. (2018). Intelligent tutoring systems for programming education: A systematic review. In *Proceedings of the 20th Australasian Computing Education Conference*, ACE '18, page 53–62, New York, NY, USA. Association for Computing Machinery.

Goode, J. (2007). If you build teachers, will students come? the role of teachers in broadening computer science learning for urban youth. *Journal of Educational Computing Research*, 36(1):65–88.

Grgurina, N., Tolboom, J., and Barendsen, E. (2018). The second decade of informatics in dutch secondary education. In Pozdniakov, S. N. and Dagienė, V., editors, *Informatics in Schools. Fundamentals of Computer Science and Software Engineering*, pages 271–282, Cham. Springer International Publishing.

Lucassen, M. (2022). Scholen schrappen vak informatica — de algemene onderwijsbond.

McVey, K. P. and Trinidad, J. (2019). Nuance in the noise: The complex reality of teacher shortages. *Bellwether Education Partners*.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3):313–350.

Shein, E. (2019). The cs teacher shortage. *Communications of the ACM*, 62(10):17–18.

Yeni, S., Aivaloglou, E., and Hermans, F. (2020). To be or not to be a teacher? exploring cs students' perceptions of a teaching career. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, Koli Calling '20, New York, NY, USA. Association for Computing Machinery.