Sebastian Hobert*

Fostering skills with chatbot-based digital tutors – training programming skills in a field study

https://doi.org/10.1515/icom-2022-0044
Received December 2, 2022; accepted May 10, 2023;
published online May 29, 2023

Abstract: Digital skills, particularly programming, have become a vital prerequisite for succeeding in today's work life. Developing those skills is, however, a challenging task, as it requires perseverance, effort, and practice. To teach coding, individualized tutoring adapted to the novice programmers' state of knowledge has evolved as the most promising learning strategy. However, offering sufficient learning support while practicing coding tasks is a challenge due to resource constraints. Following a three-cycle design science research approach, we developed a chatbot-based digital tutor that can support novice programmers using individualized, automated conversations based on adaptive learning paths and in-depth code analyses. In this article, we present the final version of the digital tutor software and report the findings of introducing it in a field setting over two entire lecture periods. We show that digital tutors can effectively provide individualized guidance in moments of need and offer high learning satisfaction in a long-term learning setting. This article expands the state of research by presenting insights into how students interact with a digital tutor over an entire lecture period. This also provides insights on how to design digital tutors for developing skills.

Keywords: chatbot; conversational agent; digital tutor; education; programming; STEM.

1 Introduction

1.1 Challenges of learning how to code

Due to digitalization, globalization, and internationalization, digital skills like programming have become a vital prerequisite for succeeding in today's work life. This great importance of coding is reflected in the fact that the European Commission has identified it as "the 21st century skill" [1]. In practice, learning how to code is, however, often considered to be a challenging task [e.g., 2, 3]: It requires perseverance and effort. Learning to code is often associated with slow progress at the beginning of the learning process but usually accelerates with more experience.

In-class teaching appears to be the standard for teaching coding skills in universities and corporate training programs. In such courses, lecturers or trainers usually present conceptual foundations of programming and the syntax of the specific programming language to be learned [4]. This is, however, insufficient. To succeed in coding, solving actual problems is required [5]. Thus, in addition to in-class teaching sessions, learners should practice exercise tasks to train to code [3, 4]. In the beginning, practicing coding by solving given exercises is, however, precisely what is often described as most challenging [4]. In the beginning stage, most learners need intense learning support and constructive feedback to solve exercise tasks appropriately. This is because determining whether an implemented algorithm solves a given problem appropriately is not trivial. To support practice, universities often offer tutorial sessions in which experienced teaching assistants support students. Due to resource constraints and increasing student numbers [6], one-on-one tutoring is usually impossible. Thus, practicing remains the most challenging part, which requires that learners persevere. Alternative teaching methods, including online-based courses such as massive open online courses (MOOCs), have similar challenges. Online training allows learners to learn the fundamentals of coding. Individual learning support while solving exercise tasks is, however, usually not available. Thus, the most challenging part of learning how to code is still unsolved.

1.2 Research objective and contribution

In this article, we address the challenge of learning how to code by introducing a digital, chatbot-based tutor that offers automated, one-on-one tutoring to novice programmers in a field setting. The digital tutor's task is to provide automated support individually to learners while practicing coding exercises. To this end, we are building on the

***Corresponding author: Sebastian Hobert**, University of Goettingen, Platz der Goettinger Sieben 5, Goettingen, 37073, Germany,
E-mail: shobert@uni-goettingen.de. https://orcid.org/0000-0003-3621-0272

emerging human-computer interaction research stream of conversational agents (CAs; also known as chatbots or smart personal assistants; see recent literature reviews [e.g., 7–9]) that have already successfully been transferred to learning settings (in this case also known as pedagogical conversational agents (PCAs); [10]). We ground this decision on the well-established theoretical foundation of the ICAP framework by [11]. The framework predicts that interactive learning engagement is superior for achieving learning success than other forms of engagement [11]. According to [11], interactive learning engagement can be reached while dialoguing or debating with others. Prior chatbot research has already shown that the technology is capable of conducting human-like conversations with users, e.g., by using so-called "social cues" [12], debating with them, or teaching them learning content to foster learning success [13]. Due to these prior research results, we developed a chatbot-based digital tutor following the design science research approach [14, 15] to transfer and adapt the technology of CA to the yet unsolved challenge of supporting learners learning how to code. In a previous work [16], we showed the results of our design and evaluation process in which we developed a conceptual artifact as well as a prototypical implementation of our digital tutor. As part of two design science research development cycles, we were already able to show that students perceive the digital tutor as useful, easy to use and assume that it might be helpful in terms of practicality. They reported a high intention to use. A limitation of this first design study was that the evaluations took place only under laboratory conditions. Therefore, students could interact with the tutor only briefly.

To address this limitation, which is typical for many design studies in the technology-enhanced research stream, our goal was to actually bring the digital tutor into teaching practice. In this article, we report the findings of finalizing the development of our digital tutor and introducing it in a field setting. This article contributes to the design knowledge base of chatbots in educational settings showing how students interact with and how they perceive the new learning experience with digital tutors in a long-term learning scenario of a full lecture period. Overall, we contribute by demonstrating that conversational educational systems are able to narrow the gap between e-learning and the gold standard of teaching—one-on-one human tutoring—by utilizing the concept of digital tutors. In addition, we contribute to general CA research by showing that CAs are not only able to converse in closely defined topic areas based on mostly predefined inputs (often utilized by FAQ bots based on static information [17] or quiz bots [e.g., 18]) but are also able to handle complex discourses with tailored responses based on dynamic, on-the-fly analyses of arbitrary inputs.

The remainder of this paper is structured as follows: First, we briefly present the theoretical background and develop two research propositions for our field study. Then, we outline the overall design science research methodology of the whole project and highlight the methodic steps of the third design and evaluation cycle targeted in this article. Subsequently, we present the final version of the digital tutor that is used productively in the field. Based on this, we describe the field setting and the evaluation results. Finally, we discuss the findings and summarize the results.

1.3 Relation to prior research activities

This paper is part of a series of projects that we have conducted since 2018 on using chatbot-based systems to support teaching and learning processes. As part of our prior research, we first conducted multiple literature-based studies on the current state of the art of using chatbots in educational settings [e.g., 10]. Afterward, we conducted several design-oriented studies in different teaching and learning settings. For instance, in [13], we conducted an experimental study in which we compared different design options of chatbots in a video-based learning setting. Two conference contributions have been published within the particular scope of the project presented in this article. First, we presented the first two intermediate design iterations of the digital tutor in [16]. Second, we discussed some data-driven results as part of a pre-study of using the digital tutor in the field at the CONVERSATIONS workshop on chatbot research in 2021 [19]. Within this article, we build on our prior contributions and extend the results substantially by presenting the final design cycle of the software implementation together with results from an in-depth field evaluation, which we conducted over the course of two full lecture periods. In doing so, we finalize the whole design science research project that lasted for approx. four years.

2 Theoretical background of the study

2.1 Digital skill adoption using conversational agent technology

Conversational agent (CA) technology is the basis for designing our human-like digital tutor in this research study. CAs

can be defined as information systems that interact with users using natural language and try to mimic a human-like conversation [20–23]. CAs are an emerging technology that dates back to the 1960s when [22] developed the first chatbot but has received increasing research interest in recent years. This is also reflected in multiple literature reviews focusing on different aspects of CAs, such as design and interaction aspects [7] or organizational settings [8, 24]. With a particular focus on educational settings [9, 10, 25], among others, present literature reviews in which they analyze how chatbots can be used in teaching and learning settings. It can be concluded that prior research proposes that CA technology is well suited for fostering learning support because it can be used to enable interactive communication between learners and a CA. It could also be shown that CAs can improve learning processes and learning outcomes in different settings.

Although we are not able to provide a complete literature review of all research on the use of educational chatbots within the scope of this paper, we would like to highlight a few relevant contributions that fit the topic as examples in addition to our own research activities (see Subsection 1.3). For a more complete overview of previous research on educational chatbots, please refer to the existing literature reviews on the topic [e.g., 9, 10, 25].

Educational chatbots have been researched in several teaching and learning settings. In this article, we focus specifically on teaching and training programming skills. With such a particular focus on supporting learning how to code a few papers have already been published: For instance [13], integrated chatbots in video-based programming training. The results reveal, for instance, that scaffolding-based conversations are superior compared to static conversation flows in terms of learning success. In another study, a chatbot supporting the teaching of python was developed by [26]. Their chatbot aims to support students while learning basic concepts (syntax and semantics) of python. The results suggest that the developed chatbot supports students in understanding python programs. In a recently published paper [27], present the results of an exploratory study with a small sample of students in which they integrated a chatbot supporting students in a python programming course. The study reports a comparably low usage, with less than 50 % of students interacting with the chatbot on average per week. The low usage rate might, for instance, be explained by the occurrence of errors in the chatbot at the beginning of the field setting. Nevertheless, the chatbot helped at least some students to learn, according to a questionnaire-based survey.

With a similar aim like the outlined prior research studies, we focus in this article on supporting novice programmers in introductory programming courses. We aim to extend the existing research results by (1) outlining how to design a chatbot-based digital tutor that is actually able to support students while working on programming exercises, and (2) by analyzing the results of a field study in which students actually interacted with the digital tutor.

From a theoretical perspective, we ground our project on the ICAP framework [11] and the scaffolding concept [e.g., 28, 29]. The ICAP framework proposes that increasing learning engagement can foster learning success [11]. According to [11], the highest form of learning engagement can be achieved in interactive learning scenarios such as conversations or discussions. This is precisely where CA technology shows its strengths as it aims to converse with users. We combine this goal of creating an engaging learning setting with the scaffolding concept [28, 29] that has previously been used in educational chatbots [13, 25]. The scaffolding concept guides our design process to provide individualized learning support based on adaptive learning paths.

2.2 Deriving research propositions from prior research

Based on prior research results summarized in the literature reviews mentioned above [e.g., 9, 10, 25], we theoretically derive research propositions. CA-based human-computer interfaces enable interactions between users and software programs in a human-like conversational way. As chatbots can be designed to foster perceived anthropomorphism [12, 30], users might perceive them as equal, human-like interlocutors. When using chatbots in educational settings, they can be perceived as having different roles, such as mentors or tutors [13, 25]. In our study, we focus on CAs that act as digital tutors, i.e., the CA takes over guidance and learning support tasks in the learning process. In this setting, the CA has at least some expert knowledge but does not teach new learning content or grade the learners. Thus, the digital tutor can support learners, e.g., while solving exercise tasks. Unlike human tutors in tutorial sessions, digital tutors can support all learners simultaneously and individually due to automatization. Thus, individualized tutoring can be enabled, which pledges effective learning support. In such a one-on-one tutoring situation, tutors can focus specifically on the learner's needs and provide individualized guidance. In individualized conversations, the digital tutor and the learner can debate the learner's solution, and an interactive learning environment can be created. According to the ICAP theory [11], positive effects on the learning outcome are to be expected. Thus, we propose the following proposition:

P1: Using digital tutors to support novice programmers in solving exercise tasks will be effective in
(a) providing guidance in moments of need, and
(b) supporting learning success.

In addition, prior research revealed that well-designed chatbots can positively affect the perceived user experience [21]. Due to the natural language communication of chatbots in a human-like way, it is generally expected that chatbots can lower usage barriers and simplify interaction. Users do not need to make themselves familiar with specific usage patterns of the user interface because they can just ask the chatbot openly for assistance. Thus, an improved user experience can be expected if the chatbot can respond automatically and individually according to the users' needs. This seems particularly true for learning settings because learners are used to asking instructors or tutors questions about specific aspects or topics (again) in detail. Even though all learning content is usually also available in other forms (such as books), discourse with instructors and tutors seems favorable, as an intense learner-to-instructor relationship can be established. The learners' specific questions and needs can be addressed in interaction with either human instructors or automated chatbots. Due to this consideration of the users' individual learning needs, an increase in perceived learning satisfaction can be expected:

P2: Using digital tutors to support novice programmers in solving exercise tasks will result in high learning satisfaction.

3 Research design

This research project follows the three-cycle design science research framework [14, 15], consisting of multiple rigor, relevance, and design cycles. During the whole search process of finding an effective artifact for the given problem [15], we conducted three design, implementation, and evaluation iterations (see Figure 1).

In the rigor cycles, we grounded our research theoretically in the ICAP framework [11] as the kernel theory and scientific literature on conversational agents and digital learning technology. Additionally, we incorporate aspects of the scaffolding concept [28, 29] in the second iteration to revise the artifact based on the first evaluation study.

As part of the relevance cycles, we conducted a qualitative workshop with experienced lecturers, e-learning professionals, and instructional designers and considered the task characteristics of teaching assistants.

During the design process, we built three artifacts: (1) a conceptual artifact visualizing the main characteristics and design features of the digital tutor, (2) a functional prototypical software artifact that implements essential features for conducting evaluation studies, and (3) a fully operational digital tutor software artifact that allows productive use in the field. To evaluate the different artifacts during the search process for a design solution, we conducted evaluation studies at the end of each artifact development phase: (1) After constructing the conceptual artifact in the first iteration, we demonstrated it to novice programmers in an introductory programming course and requested their feedback. Additionally, we asked experienced teaching assistants of multiple programming classes for suggestions for improvement. (2) After implementing the first functional prototypical software artifact in the second iteration, we introduced it to the students of an introductory programming course in a laboratory-like setting and asked them to interact with it while solving a prespecified programming task. (3) After implementing the fully operational digital tutor software artifact in the third iteration, we implemented it in an introductory programming course. These multiple evaluation studies guided our iterative search process for an effective artifact [15]. During the whole process, we followed a problem-oriented design approach [31] and took the seven guidelines for design science research [15] into account. The results of the first two iterations were communicated at the ICIS conference in late 2019 [16]. Additionally, a data-driven pre-study among a small sample from the field was discussed at the CONVERSATIONS workshop on chatbot research in 2021 [19].

In this paper, we report the final outcome and the completion of the whole project. In particular, we present the final version of the digital tutor software and report on the results of its use in a real learning setting in which $N = 155$ students actually interacted with the digital tutor.

4 The chatbot-based digital tutor

In the following, we present the final version of the chatbot-based digital tutor after the third design iteration. The presented revised concept and the final software version represent the end of the development process.

4.1 Summary of the revised concept

As outlined in the introduction section, the overall aim of the digital tutor to be developed is to offer novice programmers fully automated individualized learning support while practicing coding exercises. To achieve this automated learning support, we originally developed five design principles in the first two implementation cycles [16]. After incorporating

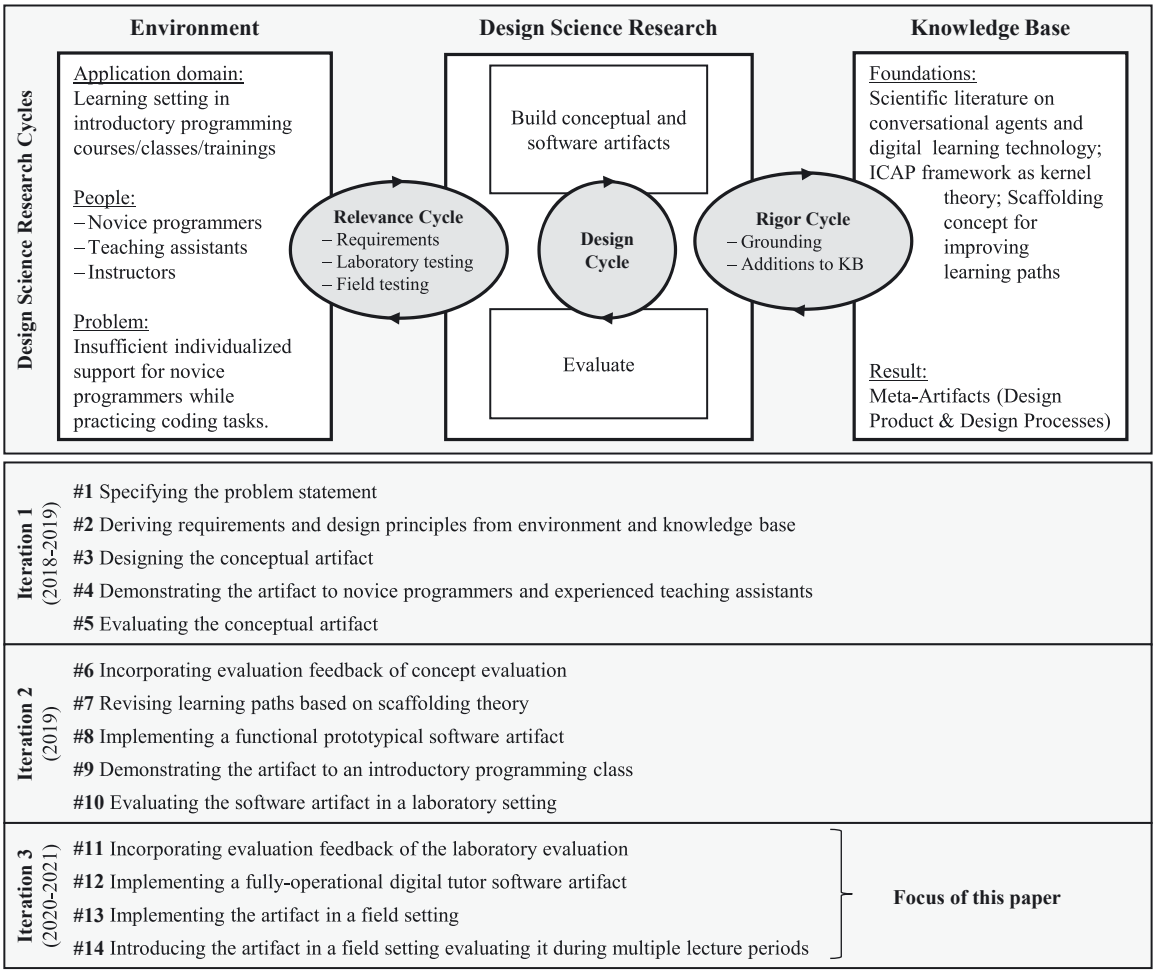

Figure 1: Overall research design based on [14–16].

the feedback of the laboratory studies, we also revised and streamlined the design principles according to the "anatomy of a design principle" by [32].

As shown in Table 1, we resulted in four design principles that summarize how developers can design a digital tutor software for fostering skills. The first design principle **DP1** focuses on fostering human-computer interaction engagement by utilizing natural language conversations. This design principle is the core principle that makes the e-learning software to be developed a digital tutor as it aims to provide a human-like interaction that imitates the interaction of students with a teaching assistant. By doing this, it is to be expected that an interactive and engaging learning environment can be created, which is expected to be beneficial for learning outcomes according to the ICAP framework [11].

The second design principle **DP2** addresses the integration of the digital tutor into the learning environment. This integration is a crucial aspect as it allows the digital tutor software to behave in an intelligent way. When supporting students while practicing coding exercises, the digital tutor needs access to the students' source code to adapt to it and provide individualized learning support. Thus, appropriate interfaces need to be integrated.

The last two design principles, **DP3** and **DP4**, focus on the actual individualization of the learning support. In particular, DP3 targets the provision of scaffolding-based learning paths and the on-demand answering of questions, while DP4 focuses on evaluating the students' input (i.e., the written source code) and deriving task-oriented feedback.

Even though we needed to adapt the design principles slightly, the corresponding system architecture remained stable after the second design cycle [16]. Thus, we did not need to make notable adjustments on a conceptual level in the third iteration. As visualized in the final version of the system architecture in Figure 2, most core components of the system architecture (like the natural language processing, dialog manager, and knowledge source) comply with the

Table 1: Revised design principles.

#	Design principle
DP1	For developers to foster novice learners' engagement in practicing skills with a chatbot-based digital tutor, ensure there is a conversational, anthropomorphic human-computer communication because this can result in human-like dialoguing [e.g., 21–23] and an interactive learning engagement, which is expected to improve learning according to the ICAP framework [11].
DP2	For developers to enable a chatbot-based digital tutor to support novice learners individually in practicing skills, ensure that all learning technologies required for practicing are integrated and interfaceable by the digital tutor because chatbot architectures require access to all necessary information to conduct individualized conversations.
DP3	For developers to provide novice learners guidance in practicing skills with a chatbot-based digital tutor, ensure that: (a) individualized adaptive learning paths can be provided, and (b) topical questions can be answered based on a sufficient knowledge base because (a) is expected to improve learning according to the scaffolding concept [e.g., 28, 29] and (b) is a main task characteristic of tutors in tutorial sessions, and learners expect such expert knowledge from tutors.
DP4	For developers to enable digital tutors to provide individualized feedback to novice learners based on their practice progress, ensure that in-depth automatic analyses of intermediate solutions can be conducted and translated into simple to understand feedback because task-oriented feedback can foster the successful completion of learning tasks [33–35].

common architecture of chatbots. Major differences exist in the user interface as it does not only consist of a natural language interface but also incorporates a source code editor, which enables students to practice coding. Additionally, the evaluation component consisting of a static and a dynamic code analyzer is added. This evaluation component makes a crucial difference as it allows the digital tutor to act in an intelligent way. Therefore, the digital tutor is not only dependent on the static knowledge source (e.g., the learning path object database) but can react dynamically to the students' individual learning process represented by the written source code.

4.2 The final version of the digital tutor software

To implement the final version of the conceptual design in our digital tutor software, we rewrote the whole software by interpreting the previous prototypes as throwaway prototypes.

The user interface shown in Figure 3 was designed as a web-based HTML5 app. To provide a clean and usable webpage experience, we used the well-established open-source AdminLTE theme [36] on top of the Bootstrap framework [37]. To integrate a source code editor, we used the

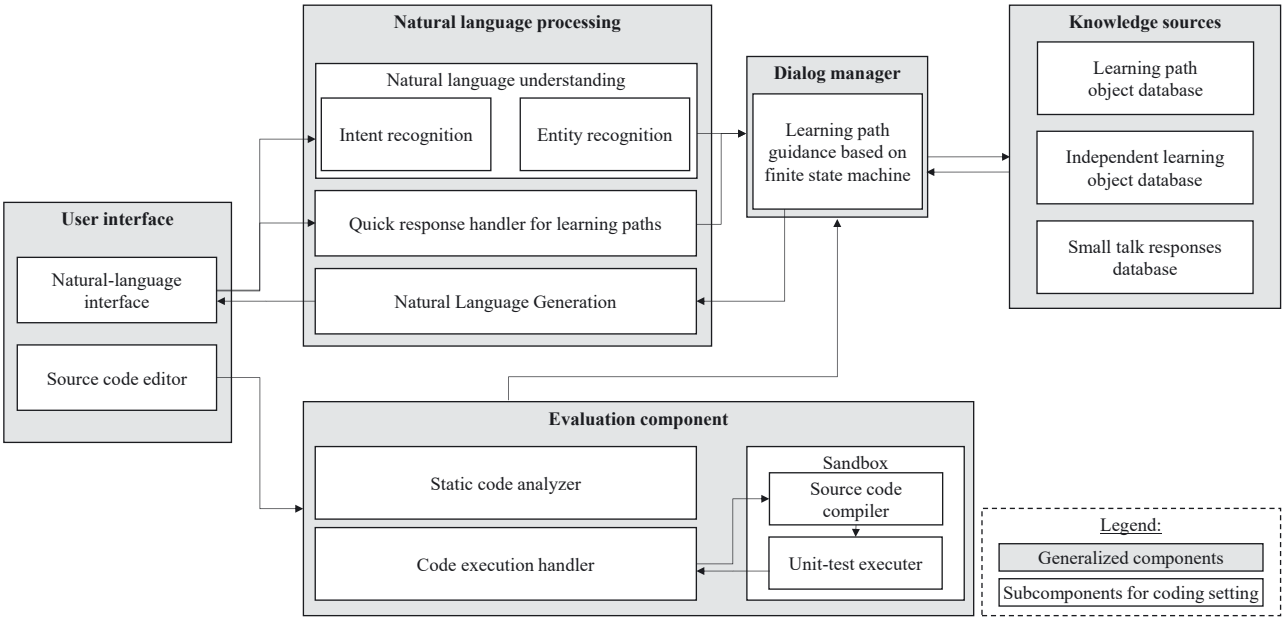

Figure 2: System architecture of the digital tutor slightly adapted from [16].

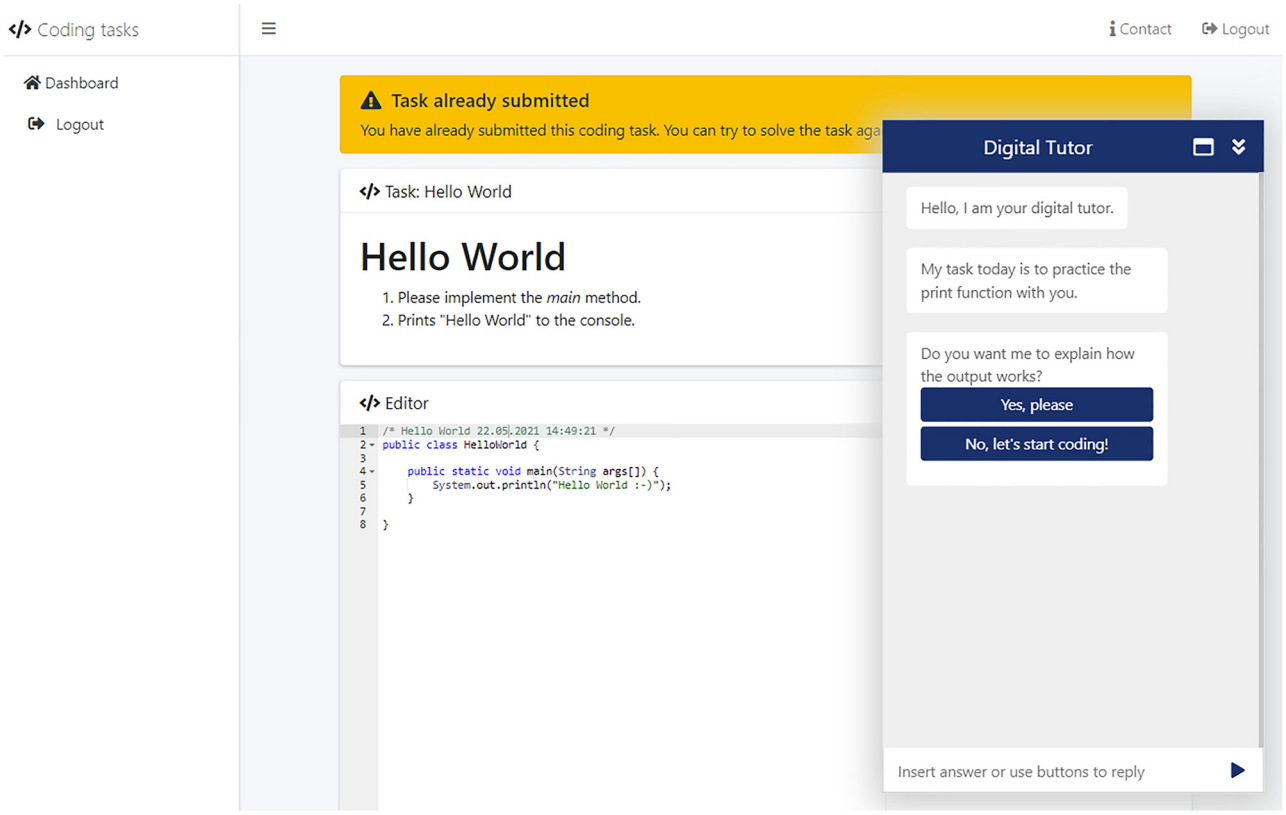

Figure 3: Screenshot of the fully operational digital tutor software artifact.

web-based ACE editor [38]. The resulting user interface consists of three major elements: (1) the exercise task description on top, (2) the source code editor as the main element, and (3) the digital tutor's chat window implemented as a dynamic popup that can be opened and closed on demand. The source code editor is the central user interface component and provides learners with typical coding environments such as syntax highlighting, autoindenting, and line numbers. The learners can insert any programming code to work on the exercise task. When the learner starts coding, the digital tutor automatically appears and starts a conversation (DP1). The digital tutor introduces itself and offers guidance (e.g., explaining the task, giving information on the theoretical background, or providing guidance through working on the solution gradually; DP3). The learner has the free choice to interact with the digital tutor or start coding without assistance. In this case, the digital tutor's popup can be closed to focus on the coding editor. Learners are free to open the digital tutor's chat popup at any time to ask for help (i.e., guidance (DP3a) or topical questions (DP3b)).

As the digital tutor can access the source code written by the learner independently of being in an active conversation (DP2), the digital tutor automatically evaluates the source code using static code analyses in its evaluation component (see system architecture; DP4). If an error repeatedly occurs during a longer timeframe (e.g., several minutes), the digital tutor will appear automatically and proactively offer the learner information on the error and further guidance. To make the proactive assistance pleasant – and not annoying or intrusive – the digital tutor points out a possible syntax error only once via a chat message. If desired, the learner can accept the offer for further help (e.g., through dynamic tests or guidance) or pause the conversation by minimizing the chat window. This proactive interaction started by the digital tutor is oriented to the behavior of human teaching assistants in programming tutorial sessions, where teaching assistants offer help to students on a voluntary base.

If the learner wants guidance, the digital tutor follows a predefined learning path, allowing adaption based on the scaffolding principle [e.g., 28, 29]. A generalized example of a learning path is visualized in Figure 4, which was the result of the first two design cycles [16]. Learning paths for the digital tutor consist of three phases that are stored in the knowledge source component of the system architecture:

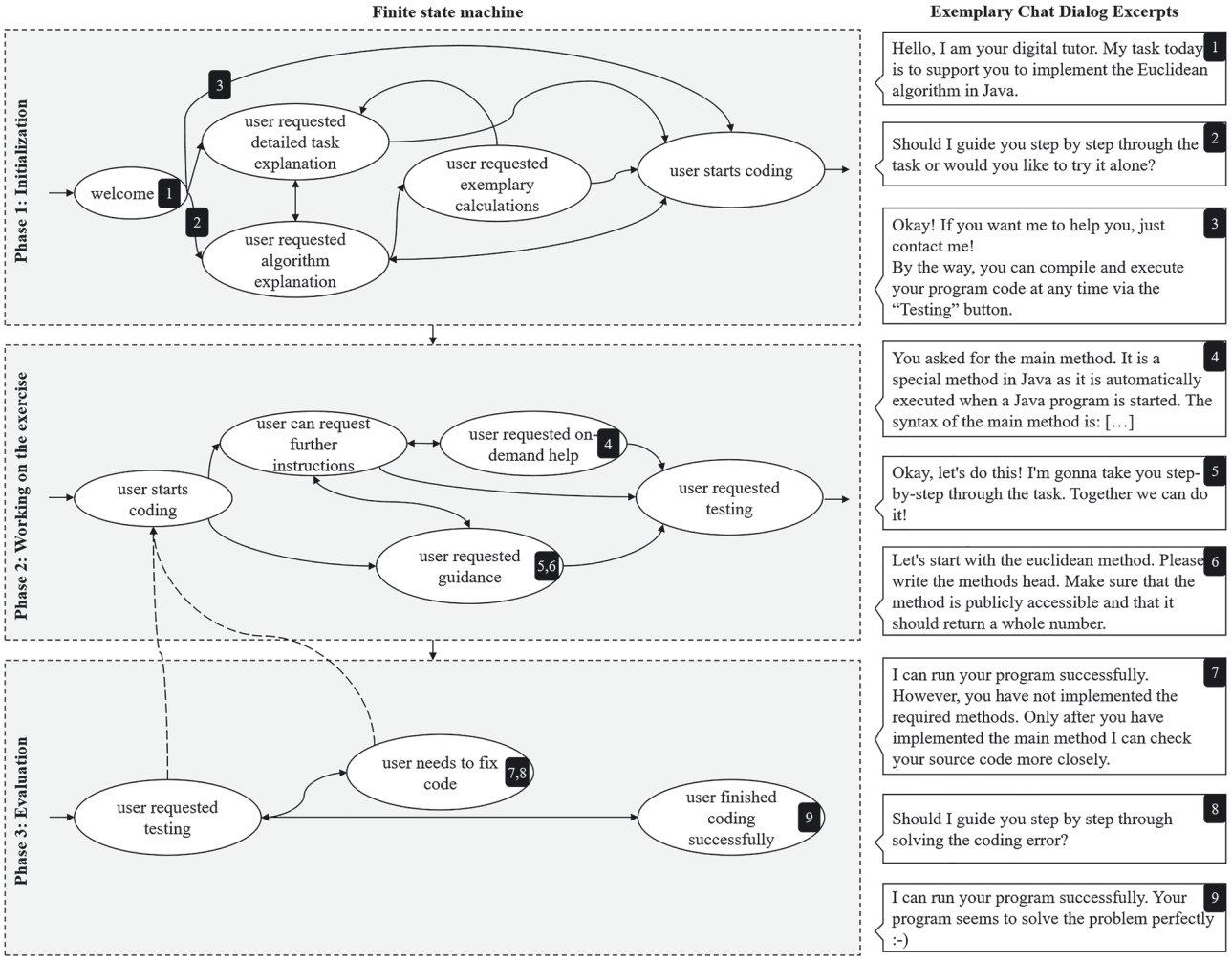

Figure 4: Visualization of learning paths using a finite state machine based on [16].

(1) The initialization phase, in which the digital tutor welcomes the learner and offers detailed explanations of the exercise task and the theoretical background. If requested, the digital tutor may exemplify the underlying algorithm using examples. Depending on the learner's choice, this information is provided in a natural language dialog, or the learner may directly start with coding without interacting in the dialog. (2) The second phase is the learner's work on the exercise. During this phase, the digital tutor may support the learner individually by answering topical questions or giving step-by-step guidance. (3) In the final phase, the learner may request testing of the worked-out solution. The digital tutor will evaluate the source code based on dynamic testing using unit tests (see system architecture; Figure 2). This allows the digital tutor to check for errors and dynamically provide individualized feedback and guidance. In the case of errors, the learner may adjust the code immediately or jump back to phase 2, in which detailed guidance is available.

The learning paths are implemented using an internal finite state machine and are hidden from the user. All information on the learning path is provided via the natural language conversation between the learner and the digital tutor.

To implement natural language communication (DP1), the messages sent by the learners are redirected to the natural language processing component, which we implemented using NLP.js [39]. The results of intent and entity recognition are handled by the dialog manager that uses different knowledge bases (i.e., learning paths (DP3a), learning objects (DP3b), and small talks (DP1)) to generate appropriate answers.

The evaluation component (DP4) encompasses two different methods for analyzing the learner's source code. First, static analyses are implemented using JavaScript to

allow instant analyses of the source code. If errors are repeatedly detected by the static analyses, the dialog manager is informed to send appropriate feedback to the learners. If learners request detailed testing, the learner's source code is sent to the dynamic testing subcomponent of the evaluation component. In this dynamic testing step, a secure sandbox is set up before the source code is compiled, executed, and tested using unit tests. Similar to the results of the static tests, the results of the analyses are sent to the dialog manager.

The resulting final version of the digital tutor was deployed on multiple virtual machines/containers to separate the user interface and the knowledge databases from the dynamic source code analyzer. This adds an additional layer of security over the used sandboxing environment and allows to dynamically adapt to load peaks which might occur when an unusually large number of students want to run and test their source code at the same time. Due to this deployment setup, we could solve the only identified performance issue that occurred during the second laboratory test in cycle 2 [16]. With this new deployment infrastructure, we did not run into any performance issues during the field study.

5 Evaluation in the field

Following the typical design-build-evaluate procedure in design science research [40], we conducted one demonstration and evaluation study after each design phase. In the first design iteration, we focused on getting feedback on how to improve the conceptual design of the digital tutor [16]. Subsequently, students interacted with a functional prototype in a laboratory setting on average for approximately 20 min at the end of the second design cycle [16]. Even though the interaction time was too short in the first two evaluations to get actual insights into how students interact with a digital tutor in the long term, we were able to get first positive insights into the students' perceived user experience and retrieved qualitative suggestions for improving the software. Nevertheless, getting deep insights into how students interact with and perceive the learning experience with such a digital tutor in the long term of a full lecture period was not possible in the laboratory setting. Thus, we outline the results of using the improved productive version of the digital tutor in the following.

5.1 Evaluation setting

After finalizing the digital tutor software in the third evaluation cycle (see Section 4.2), we introduced it in a programming course targeting novice programmers in two identical lecture periods. At the beginning of the course in each lecture period, we asked the novice programmers to participate in our final evaluation study on a voluntary basis following the institutional guidelines. We provided an access code to all students enrolled in the course and asked for informed consent to participate in the study. Participation in the research activities was completely voluntary and no incentives were offered. Students were free to choose whether to participate in research activities regardless of whether they used the digital tutor. A total of $N = 155$ students agreed to participate and interacted with the digital tutor. In the first week of each lecture period, we asked those students whether they had any prior programming experience. On a Likert scale ranging from -3 (no programming experience) to $+3$ (expert knowledge), the resulting average of -1.024 confirms that the course targets novice programmers. During the subsequent months, the students interacted with the digital tutor to work on programming tasks.

During the semesters, we provided two to three weekly exercises. The students had the opportunity to work on the exercises, discourse with the digital tutor, and obtain automatic feedback using the digital tutor's static and dynamic code analyses. For those students who voluntarily agreed to participate in the study, we collected usage statistics of the interaction with the digital tutor and asked each student to rate the difficulty of each exercise task. At the end of each week, a (human) tutor manually evaluated the students' assignments and graded them on a scale from 0 to 10.

At the end of the lecture term, we asked the participating students to provide us feedback on their perceived learning satisfaction with the digital tutor, user experience based on the UEQ+ scale [41, 42], and the evaluation of our specific design principles as outlined in the following. Figure 5 summarizes the study design.

5.2 Evaluation results of interactions based on usage data

During the field study, the students interacted with the digital tutor in approximately 94,000 chat messages, and the digital tutor conducted approximately 29,000 dynamic tests to provide individualized feedback on the students' source code (code executions and unit tests).

An analysis of the usage times in which the students interacted with the digital tutor in the chat clearly shows that students distributed their learning activities over the entire week (see Figure 6). The students interacted with the digital tutor on every day of the week and on every hour of the day. A peak value of approx. 32 % can be determined

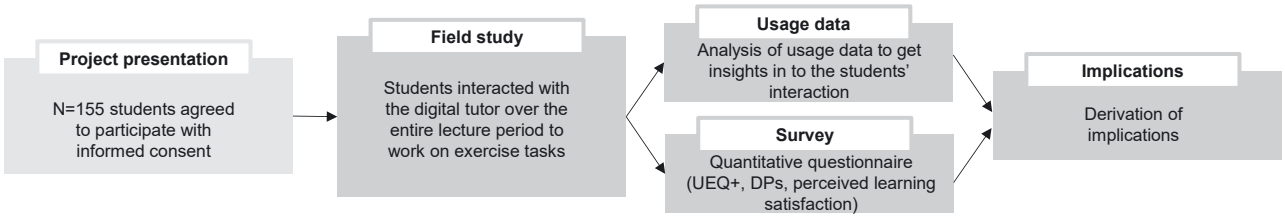

Figure 5: Overview of field study.

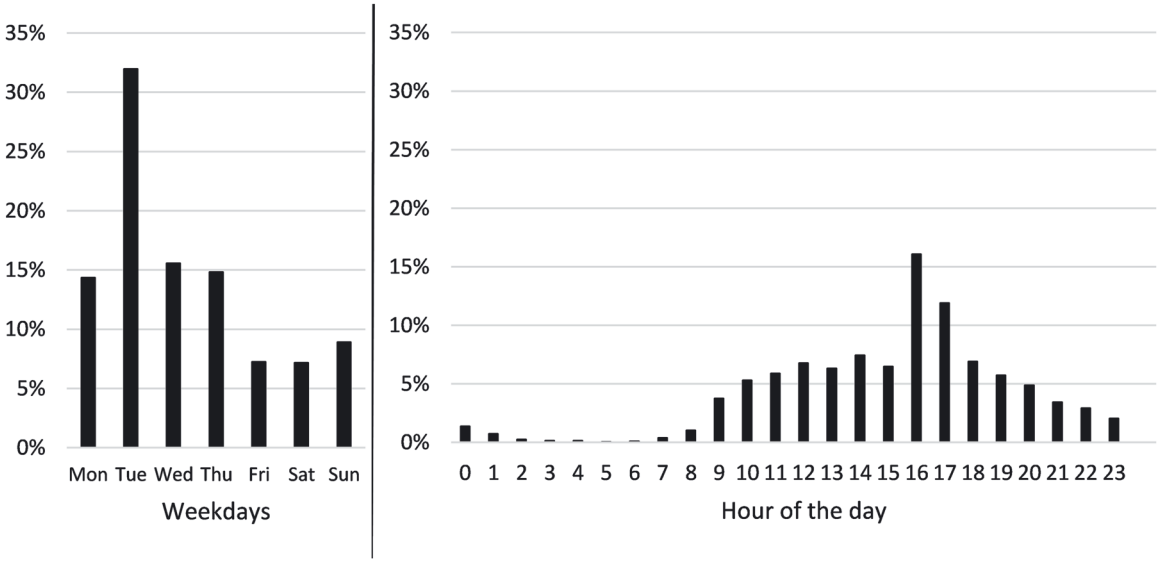

Figure 6: Visualization of chat interactions over time (per weekday resp. per hour of the day).

on Tuesdays. This is not surprising as a tutorial session was offered each week on Tuesdays in which a teaching assistant was available for help. Nevertheless, the remaining learning activity distributes among the other days with values between approx. 7 %–16 %. A similar pattern can be observed when analyzing the interaction times per hour of the day. We can see a peak between 4 pm and 5 pm. This matches the time of the tutorial session on Tuesdays. During the remaining hours of the day, remarkable interaction activities can be seen between 9 am and 11 pm. Only during the night, very few interactions took place.

This analysis of the interaction times by weekdays and hours of the day clearly shows that the students' learning activities are not limited to typical office hours in which human tutors would be available (e.g., in our case: Tuesdays from 4 pm to 6 pm). Instead, the analysis reveals that students distribute their learning activities over the entire week. This analysis provides first evidence that the digital tutor was able to support students in moments of need, i.e., when they actually train their programming skills.

To further analyze whether the digital tutor provides guidance in moments of need (i.e., in situations when students perceive an exercise task as challenging), we analyzed the difficulty of each exercise task based on the students' responses and related it to the number of associated chat interactions and dynamic tests. As outlined in Figure 7 (top), a strong correlation [43] exists between both and the average perceived difficulty of the exercises ($r_{s\_chat} = 0.798$ resp. $r_{s\_test} = 0.700$). This provides evidence that the interaction with the digital tutor is increased when students are working on complex and difficult tasks. This further supports our proposition that digital tutors are particularly helpful in moments of need. Furthermore, this indicates that students are not always interacting with the digital tutor to the largest possible extent (which would mean that they always ask for guidance while solving the tasks). Instead, they work more independently to solve the problem when the exercises are perceived as less difficult.

One additional interesting observation is that there are noticeable differences in the number of chat interactions and tests performed for exercises with similar perceived difficulty. This can be seen, for instance, in the exercises

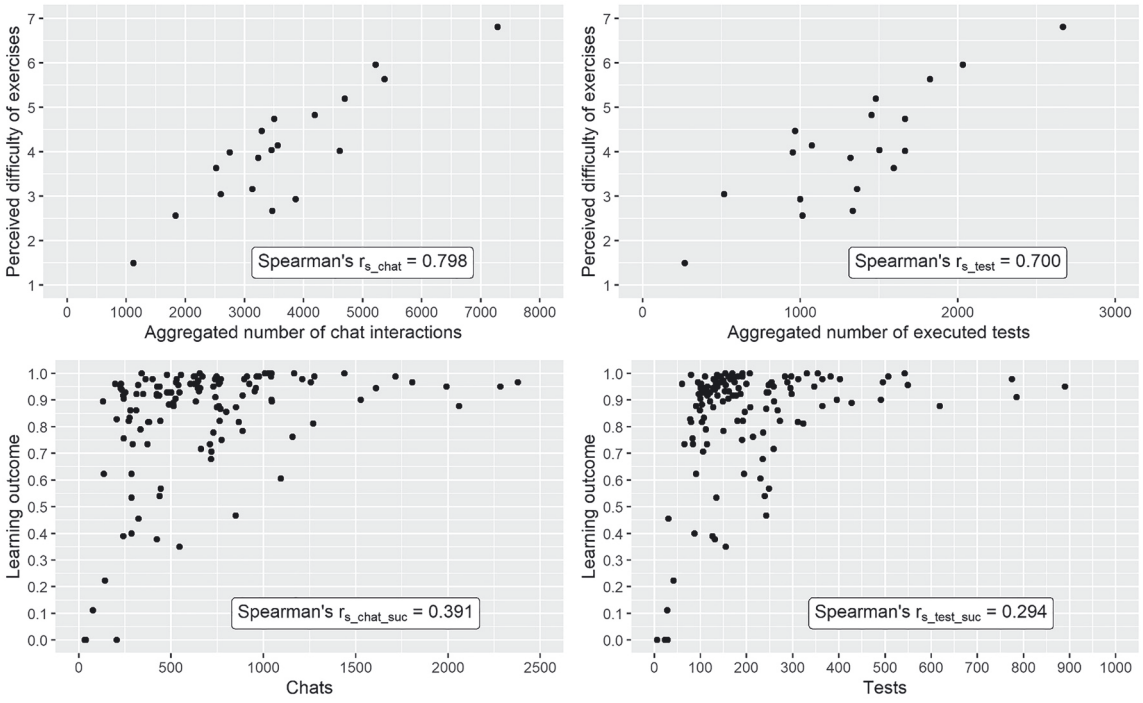

Figure 7: Visualization of chat interactions and executed tests in relation to the difficulty of exercises.

that were rated with a difficulty of approximately 4. In these cases, the number of chat interactions ranges from less than 2800 to more than 4600 interactions. At the same time, the interval for the tests performed ranges from less than 1000 to more than 1600. A closer look shows that out of these five exercises with a difficulty of approximately 4, the exercise with the highest number of chat interactions also has the highest number of tests. This exercise task was given at the beginning of the lecture period. For exercises with a comparable difficulty given later in the semester, the number of interactions seems to be lower. Possible explanations could be, for instance, that (a) the students wanted to try out the guidance functionalities of the digital tutor more often at the beginning of the semester or (b) that they felt more confident in programming as the semester progressed. Further investigations are needed here in future research.

When analyzing the students' individual learning outcomes (see Figure 7 at the bottom), (almost) moderate correlations were found between chats and tests ($r_{\text{s\_chat\_suc}} = 0.391$ resp. $r_{\text{s\_test\_suc}} = 0.294$). Regarding learning success, the manual grading of the teaching assistant shows that students performed as desired in the exercise tasks. The manual grading resulted in an average learning outcome rate of 84 % (i.e., the participating students achieved, on average, 84 % of all points in all exercises). Even though we could not set up a control group in our field study due to ethical considerations (see discussion of limitations in Subsection 6.3), we can report that the resulting learning outcome was better than expected and exceeded the results from previous lecture terms.

Overall, the analyses of the usage data indicate that the digital tutor is capable of (a) providing appropriate guidance for students when they need assistance, and (b) we assume that it supports learning success. To support this conclusion further, we analyze the survey results below.

5.3 Evaluation results of the field study based on survey data

To obtain further evidence of whether the digital tutor actually supports novice programmers in solving exercise tasks (research proposition P1), we analyze the students' self-reflection based on the survey data on a Likert scale ranging from −3 (not useful) to +3 (very useful).

The students reported that, overall, they were satisfied with the user experience of the digital tutor. According to the survey results using the UEQ+ scale [41, 42], which extends the well-established user experience questionnaire [44, 45] modularly by additional scales, the digital tutor artifact received positive evaluation results with average values larger than +1.00. As shown in Figure 8, the usefulness of the digital tutor received the most positive feedback, whereas the response quality of the digital tutor's chat interaction received lower but still positive ratings. Even the lowest

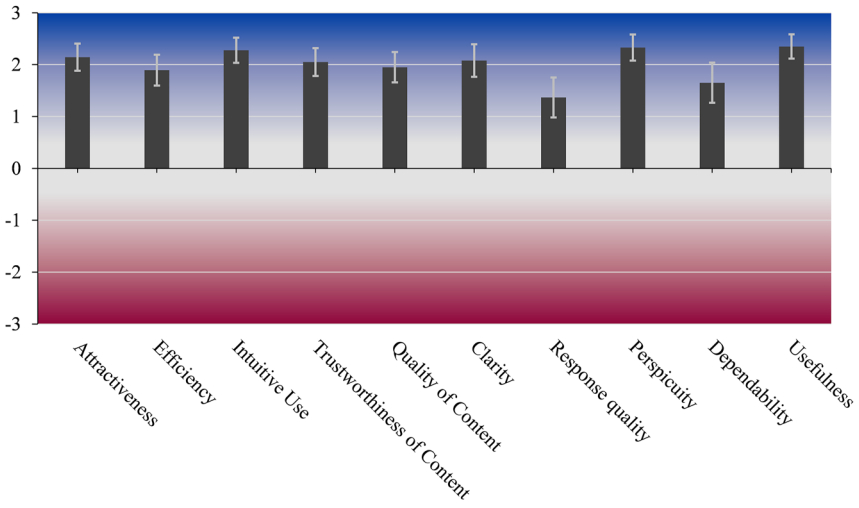

Figure 8: Visualization of the results of the UEQ+ based evaluation based on the official *Data Analysis Tool* [41].

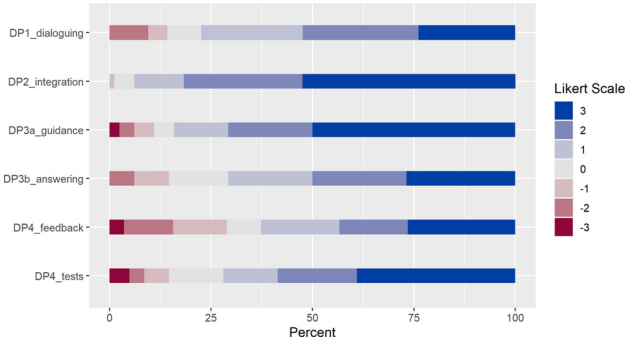

Figure 9: Visualization of the evaluation of the functionalities grounded in the design principles.

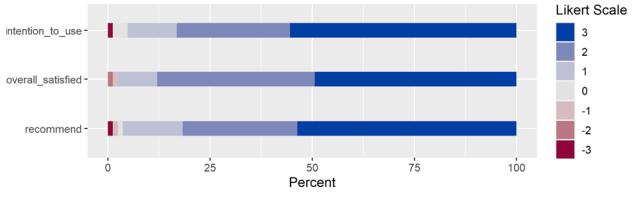

Figure 10: Visualization of the overall evaluation.

value of +1.37 represents a positive result in terms of user experience. From this user experience evaluation, the positive results of our previous studies can be confirmed.

We also focused on the students' evaluation of the specific functionalities of the digital tutor in this third evaluation study (see Figure 9). To this end, we focus on the four design principles outlined in Section 4. In particular, the students value the digital tutor's capability to engage in dialogues (DP1, avg. + 1.298) and the integrated platform (DP2, avg. + 2.268) that combines all e-learning components required for working on the coding tasks (editor, chat interface, automatic testing, guidance, feedback). The specific functionalities to support the students' learning processes also received promising feedback: individualized guidance based on learning paths (DP3a, avg. + 1.854) and the digital tutor's capability of answering individual questions (DP3b, avg. + 1.268) were both rated positively. Additionally, the students evaluated the individualized feedback (avg. + 0.843) based on automatic tests (avg. + 1.415) as helpful.

Finally, we asked the students to also reflect on their overall learning satisfaction (see Figure 10). The students reported learning satisfaction with an average value of +2.313 as high. Consequently, the students also intend to use the platform in the future (if available in further courses; avg. +2.301) and recommend it to their fellow students (avg. +2.268).

6 Discussion

In our design science research study, we developed a new chatbot-based digital tutor software for the educational context. The digital tutor's aim is to support students while training programming skills offering automatic and individual help using chat-based guidance, automatic feedback, and on-demand dialoguing capabilities. With our iterative design-build-evaluate procedure [40] of the three-cycle design science information systems research framework [14, 15], we strove to find an effective software artifact [15] for the given problem of supporting the challenging learning process of novice programmers while training how to code using exercise tasks. We developed one conceptual artifact and two software artifacts during the scientific

search process of finding a suitable solution. We demonstrated the three artifacts in two laboratory evaluation studies [16] and one field study to more than 200 students in total.

In all three evaluation studies, we revealed that students welcome the digital tutor concept and consider the automatic and individualized learning support provided by an intelligent digital tutor as helpful and desirable. We recorded this positive perception already in the first evaluation study. In the two design-build-evaluate iterations [40] that followed, we refined the concept gradually based on the evaluation results. At the end of the third iteration and as a result of this article, we result in a productive digital tutor software (see Section 4.2).

Our field study with 155 participating students in a programming course targeting novice programmers revealed that the students were satisfied with the digital tutor software, and we could not identify any serious issues related to the artifact design. Additionally, our analysis of the students' interactions with the digital tutor demonstrates first evidence that the digital tutor reached its overarching goal of supporting the learning processes of novice programmers. According to the evaluation results from the field, we expect that the research propositions hold. Thus, we have reasonable grounds to believe that we successfully found and designed a useful software for the given research problem. For this reason, we believe we can complete this extensive design project after four years with confidence. With these results, we intend to provide prescriptive knowledge with our design principles that describe how to implement digital tutors for skill training [46].

Overall, with the contribution of this article in terms of providing a productive digital tutor software and getting insights into how students interact with a digital tutor in the field, we extend our prior research activities (see Subsection 1.3) substantially. As we didn't encounter severe issues of the digital tutor software during the field test and the field evaluations indicate that the digital tutor provides added value students, we end our design science research project at this point. Nevertheless, besides this design science research project which mainly focuses on generating design knowledge, future research is needed to fully investigate how digital tutors can support students and which effects the use of digital tutors have in learning settings (see Subsection 6.3).

6.1 Implications for developing programming skills

The results of the analysis of the field study have shown that students interact with the digital tutor over the entire week, on every day of the week, and on every hour of the day. This clearly shows that the digital tutor is superior in terms of availability compared to the typical working hours of human tutors.

Interestingly is the finding that the students increase their interaction with the digital tutor when the exercises become more complex. This is in line with our argument that coding is a challenging task, as described in the introduction section [e.g., 2, 3], and with our data-driven prestudy among a small sample [19]. The perceived difficulty of a coding exercise can have multiple reasons. According to our teaching experience, reasons for the perceived difficulty of learning how to code might be (1) the need to understand and break down the problem to be solved, (2) the construction of an algorithm suited for the given exercise task, (3) the transformation of the identified algorithm into actual source code, and (4) the evaluation of the suitability of the written code. In those four aspects, our designed digital tutor can support novice programmers: (1 + 2) The digital tutor can explain the theoretical background of coding tasks based on individual learning paths (DP3a), (3) the digital tutor is capable of answering questions related to coding (DP3b) and providing guidance for solving the tasks (DP3a), and (4) the digital tutor can provide the novice programmers feedback based on automatic assessments using unit tests and static code analysis (DP4). The results of our evaluation studies indicate the success of this multidimensional learning support strategy. The survey results show that the digital tutor's capabilities (grounded in the design principles) are reported to be helpful for the participating novice programmers. We could further show that there exists a correlation between the number of interactions and learning success. In addition to our subjective impression of better learning outcomes compared to previous lecture terms, this is a further indicator that the digital tutor's capabilities support learning success. However, to actually measure the effect, future research with a control group design is needed (see the discussion of limitations in Subsection 6.3). Nevertheless, based on our insights, we expect that research proposition P1 holds:

P1: Using digital tutors to support novice programmers in solving exercise tasks will be effective in (a) providing guidance in moments of need and (b) supporting learning success.

The high number of interactions of the novice programmers with the digital tutor already suggested that the students value the digital tutor's learning support, which is supported by the positive evaluation in the survey. The students reported extraordinarily high learning satisfaction after interacting with the digital tutor for several months in

the field. Thus, we expect that research proposition P2 can be confirmed in our field setting:

P2: Using digital tutors to support novice programmers in solving exercise tasks will result in high learning satisfaction.

Based on the results of our study, we can conclude that training programming skills can successfully be supported using chatbot-based educational technologies. Even when human resources are limited (e.g., due to resource constraints of institutions), digital tutors can help to close the gap. Digital tutors can be designed based on the four design principles to assist tasks typically executed by teaching assistants.

6.2 Implications and possibilities for transfer

Even though we focus on programming skills—one of the most important 21st century skills [1, 5, 47]—we argue that our results are also transferable to other learning settings. Our research project showed that digital tutors could be conceptualized and built so that complex learning settings can be supported. To this end, we showed that digital tutors—based on chatbot technology—can support learning processes. This is in line with prior research in which it has been shown that scaffolding-based CAs are capable of supporting learning success [e.g., 13]. In a substantial improvement over the prior state of the art, we showed that a digital tutor could combine multiple technologies to support learning processes. In our case, we combined (1) CA technology with a finite state machine to provide learners scaffolding-based guidance and on-demand help, (2) static and dynamic code analysis technology known from software testing, and (3) a web-based programming environment. This combination of different technologies enables a digital tutor to obtain deep insights into learning processes and enables starting points for automatic, individualized support.

This overarching architecture that combines CAs with guidance and an evaluation component enables a transfer to other skill adoption settings. For instance, the concept could be transferred to other skill adoption settings that encompass problem-solving. Exemplary settings that are, in our opinion, easy to realize based on our design principles and system architecture can be found in the disciplines of data science (e.g., learning statistical analyses or machine learning topics), computer science (e.g., learning modeling as part of software engineering), or mathematics (e.g., learning to calculate). In those learning scenarios, the learning content, the learning paths, and the evaluation component need to be adapted as well as the user interface. However, the overall system architecture of the digital tutor (see Subsection 4.1) could remain the same. These related learning scenarios offer starting points for future research.

A more specific transfer example of a potential learning setting to which the digital tutor concept could be transferred to is supporting modeling exercises in software engineering use cases (e.g., event-driven process chains). Here, learners could be supported step-by-step while working on a modeling task as part of a case study. Using automatic analysis, a digital tutor could check for possible formal errors in the created model (e.g., correct formal use of events and functions). In addition, the modeling solution could also be automatically checked for the correctness of the content. However, designing a fully automated test procedure (similar to dynamic tests known from programming) could be challenging if the names of events and functions are not chosen uniformly.

Apart from the learning settings, we were able to show that chatbot technology does not have to be limited to more simple use cases in which a chatbot merely answers basic questions based on a predefined knowledge base. We showed that chatbots could integrate complex evaluation procedures to make them more *intelligent*. In our case, the evaluation components make a substantial difference compared to simple question-and-answer chatbots. Here, automatic evaluation can enable individualized feedback, guidance, and helpful learning support. This is a suggestion for chatbot projects in future research so that smarter chatbots can emerge in other application domains as well, combining automated analytics with process support.

6.3 Limitations and future research

In this design science research project, we followed an iterative development process based on the typical design-build-evaluate procedure [40]. We followed the three-cycle design science research framework [14, 15] by grounding our research on scientific theories (such as the ICAP framework [11] and the scaffolding principle [e.g., 28, 29]) and on insights from practice (e.g., by conducting an expert workshop and analyzing typical learning settings). Despite this structured and rigorous research process, the individual studies we conducted during our whole research activities might have limitations. For instance, the selection of the field setting at our university and the sample of novice programmers who agreed to participate in our field study might have influenced the results. Therefore, conducting further evaluation studies based on our design principles might help to validate the results.

Additionally, a critical point in many evaluation studies in educational contexts is the analysis of learning success. To

obtain initial insights into the impact of the digital tutor on learning success, we combined the analyses of the students' interactions with the digital tutor and a survey in which students reported their self-perception of the learning process. This enables us to obtain insights into the effects on learning success. Nevertheless, we were not able to apply an experimental study design due to ethical considerations. If we had withheld the digital tutor from a subsample (e.g., control group) of the participants, we expect that those students would have been disadvantaged. Such a control group design can obviously not be implemented in a real learning setting in the field if it can be assumed that the software under investigation offers a strong positive added value to the students' learning process. Choosing a laboratory study design (as we did in the first two evaluation studies in [16]) would not be helpful either, as we would not be able to cover a long-term learning scenario. Thus, we believe that our single-group field study design is an appropriate methodic approach to evaluate the success of the productive version of the digital tutor. Nevertheless, in the future, further (laboratory) studies with a control group design would be valuable to actually measure the effect size of digital tutors on the learning outcome. Here, an experimental setting with (at least) two groups of students would be particularly interesting. It should be ensured that both groups receive the same learning content and participate in identical formative assessments in a long-term learning setting. However, only one group of students should have access to the digital tutor. By comparing the learning outcome, it would be possible to analyze the actual impact on the learning process. This could provide further evidence of the effectiveness of digital tutors for the development of skills.

7 Conclusions

Our extensive design science research project and the field study, in particular, show that digital tutors can support skill adoption processes. In particular, we offer a solution that supports the students' programming skill adoption by combining individualized guidance based on extensive learning paths and knowledge bases with automatic feedback provision based on dynamic and static code analyses. With the four provided design principles, we present prescriptive design knowledge. Our empirical results from the field provide evidence that digital tutors can support novice programmers successfully when learning how to code. We hope our results (1) inform the design of future digital tutors to improve learning support and increase learning satisfaction in current and future educational settings and (2) support general CA design research with our presented system architecture and software implementation.

Acknowledgment: We would like to thank the students who supported this project by participating in the research activities.
Author contribution: The author has accepted responsibility for the entire content of this submitted manuscript and approved submission.
Research funding: The transfer of our research findings to teaching practice has been supported by the Niedersächsisches Ministerium für Wissenschaft und Kultur.
Conflict of interest statement: The author declares no conflicts of interest regarding this article.

References

1. European Commission. *Coding - the 21st Century Skill - Shaping Europe's Digital Future - European Commission*, 2021. https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill (accessed Aug 6, 2021).
2. Daradoumis T., Marquès Puig J. M., Arguedas M., Calvet Liñan L. Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming. *Comput. Educ.* 2019, *128*, 159—170.
3. Vial G., Negoita B. Teaching programming to non-programmers: the case of Python and jupyter notebooks. In *ICIS 2018 Proceedings*, 2018; pp. 1—17.
4. Passier H. The role of procedural guidance in software engineering education. In *Proceedings of the International Conference on the Art, Science, and Engineering of Programming - Programming '17*, 2017; pp. 1—2.
5. Nouri J., Zhang L., Mannila L., Norén E. Development of computational thinking, digital competence and 21 st century skills when learning programming in K-9. *Educ. Inq.* 2020, *11*, 1—17.
6. Maedche A., Legner C., Benlian A., Berger B., Gimpel H., Hess T., Hinz O., Morana S., Söllner M. AI-based digital assistants. *Bus. Inf. Syst. Eng.* 2019, *61*, 535—544.
7. Diederich S., Brendel A., Morana S., Kolbe L. On the design of and interaction with conversational agents: an organizing and assessing review of human-computer interaction research. *J. Assoc. Inf. Syst.* 2022, *23*, 96—138.
8. Meyer von Wolff R., Hobert S., Schumann M. How may I help you? — State of the art and open research questions for chatbots at the digital workplace. In *Proceedings of the 52th Hawaii International Conference on System Sciences*, 2019; pp. 95—104.
9. Winkler R., Söllner M. Unleashing the potential of chatbots in education: a state-of-the-art analysis. In *Academy of Management Annual Meeting (AOM)*, 2018.
10. Hobert S., Meyer von Wolff R. Say hello to your new automated tutor — a structured literature review on pedagogical conversational agents. In *Proceedings of the 14th International Conference on Wirtschaftsinformatik*, 2019; pp. 301—314.

11. Chi M. T. H., Wylie R. The ICAP framework: linking cognitive engagement to active learning outcomes. *Educ. Psychol.* 2014, *49*, 219–243.
12. Feine J., Gnewuch U., Morana S., Maedche A. A taxonomy of social cues for conversational agents. *Int. J. Hum. Comput. Stud.* 2019, *132*, 138–161.
13. Winkler R., Hobert S., Salovaara A., Söllner M., Leimeister J. M. Sara, the lecturer: improving learning in online education with a scaffolding-based conversational agent. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020; pp. 1–14.
14. Hevner A. A three cycle view of design science research. *Scand. J. Inf. Syst.* 2007, *19*, 87–92.
15. Hevner A., March S., Park J., Ram S. Design science in information systems research. *Manag. Inf. Syst. Q.* 2004, *28*, 75–105.
16. Hobert S. Say hello to 'coding tutor'! Design and evaluation of a chatbot-based learning system supporting students to learn to program. In *ICIS 2019 Proceedings*, 2019; pp. 1–17.
17. Meyer von Wolff R., Hobert S., Masuch K., Schumann M. Chatbots at digital workplaces - a grounded-theory approach for surveying application areas and objectives. *Pac. Asia J. Assoc. Inf. Syst.* 2020, *12*, 64–102.
18. Ruan S., Jiang L., Xu J., Tham B. J.-K., Qiu Z., Zhu Y., Murnane E. L., Brunskill E., Landay J. A. QuizBot: a dialogue-based adaptive learning system for factual knowledge. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019; pp. 1–13.
19. Hobert S. Individualized learning patterns require individualized conversations — data-driven insights from the field on how chatbots instruct students in solving exercises. In *Chatbot Research and Design*; Følstad A., Araujo T., Papadopoulos S., Law E. L.-C., Luger E., Goodwin M., Brandtzaeg P. B., Eds. Springer International Publishing, 2022; pp. 55–69.
20. Brandtzaeg P. B., Følstad A. Chatbots: changing user needs and motivations. *Interactions* 2018, *25*, 38–43.
21. Følstad A., Brandtzaeg P. B. Users' experiences with chatbots: findings from a questionnaire study. *Qual. User. Exp.* 2020, *5*, 1–14.
22. Weizenbaum J. ELIZA - a computer program for the study of natural language communication between man and machine. *Commun. ACM* 1966, *9*, 36–45.
23. Diederich S., Brendel A. B., Kolbe L. M. Designing anthropomorphic enterprise conversational agents. *Bus. Inf. Syst. Eng.* 2020, *62*, 193–209.
24. Lewandowski T., Delling J., Grotherr C., Böhmann T. State-of-the-Art analysis of adopting AI-based conversational agents in organizations: a systematic literature review. In *PACIS 2021 Proceedings*, 2021; pp. 1–14.
25. Wollny S., Schneider J., Di Mitri D., Weidlich J., Rittberger M., Drachsler H. Are we there yet? - a systematic literature review on chatbots in education. *Front. Artif. Intell.* 2021, *4*, 654924.
26. Chinedu O., Ade-Ibijola A. Python-bot: a chatbot for teaching Python programming. *Eng. Lett.* 2021, *29*, 25–34.
27. Carreira G., Silva L., Mendes A. J., Oliveira H. G. Pyo, a chatbot assistant for introductory programming students. In *2022 International Symposium on Computers in Education (SIIE)*, 2022; pp. 1–6.
28. Kim M. C., Hannafin M. J. Scaffolding problem solving in technology-enhanced learning environments (TELEs): bridging research and theory with practice. *Comput. Educ.* 2011, *56*, 403–417.
29. van de Pol J., Volman M., Beishuizen J. Scaffolding in teacher–student interaction: a decade of research. *Educ. Psychol. Rev.* 2010, *22*, 271–296.
30. Seeger A.-M., Pfeiffer J., Heinzl A. Texting with human-like conversational agents: designing for anthropomorphism. *J. Assoc. Inf. Syst.* 2021, *22*, 931–967.
31. Peffers K. E., Tuunanen T., Rothenberger M. A., Chatterjee S. A design science research methodology for information systems research. *J. Manag. Inf. Syst.* 2008, *24*, 45–77.
32. Gregor S., Kruse L., Seidel S. Research perspectives: the anatomy of a design principle. *JAIS* 2020, *21*, 1622–1652.
33. Hattie J., Timperley H. The power of feedback. *Rev. Educ. Res.* 2007, *77*, 81–112.
34. Piccoli G., Rodriguez J., Palese B., Bartosiak M. L. Feedback at scale: designing for accurate and timely practical digital skills evaluation. *Eur. J. Inf. Syst.* 2020, *29*, 114–133.
35. Kluger A. N., DeNisi A. The effects of feedback interventions on performance: a historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychol. Bull.* 1996, *119*, 254–284.
36. AdminLTE.io. *ColorlibHQ/AdminLTE*, 2020. https://github.com/ColorlibHQ/AdminLTE (accessed Feb 16, 2020).
37. Otto M., Thornton J., Bootstrap contributors. *Bootstrap-The Most Popular HTML, CSS, and JS Library in the World*, 2021. https://getbootstrap.com/ (accessed Jun 25, 2021).
38. ACE. *Ace - The High Performance Code Editor for the Web*, 2021. https://ace.c9.io/ (accessed June 25, 2021).
39. AXA Group Operations Spain S.A. *axa-group/nlp.js*, 2023. https://github.com/axa-group/nlp.js (accessed May 22, 2023).
40. March S. T., Smith G. F. Design and natural science research on information technology. *Decis. Support Syst.* 1995, *15*, 251–266.
41. UEQ+ Team. *UEQ+. A Modular Extension of the User Experience Questionnaire*, 2021. https://ueqplus.ueq-research.org/ (accessed on July 20, 2021).
42. Schrepp M., Thomaschewski J. Design and validation of a framework for the creation of user experience questionnaires. *IJIMAI* 2019, *5*, 88–95.
43. Cohen J. *Statistical Power Analysis for the Behavioral Sciences*; Lawrence Erlbaum Associates: Hillsdale, NJ, 1988.
44. Laugwitz B., Held T., Schrepp M. Construction and evaluation of a user experience questionnaire. In *HCI and Usability for Education and Work*; Holzinger A., Ed., Springer: Berlin, Heidelberg, 2008, pp. 63–76.
45. Team U. E. Q. *User Experience Questionnaire (UEQ)*, 2020. https://www.ueq-online.org/ (accessed Dec 14, 2020).
46. Gregor S., Hevner A. R. Positioning and presenting design science research for maximum impact. *MIS Quarterly* 2013, *37*, 337–356.
47. Popat S., Starkey L. Learning to code or coding to learn? A systematic review. *Comput. Educ.* 2019, *128*, 365–376.

Bionotes

Sebastian Hobert
University of Goettingen, Platz der Goettinger Sieben 5, Goettingen, 37073, Germany
shobert@uni-goettingen.de
https://orcid.org/0000-0003-3621-0272

Sebastian Hobert is a postdoctoral researcher at the University of Goettingen. He works in the interdisciplinary field of information systems, computer science, and human-computer interaction. His research mainly focuses on technology-enhanced learning, including chatbot-based digital tutors, learning analytics, and AI-based methods to support teaching and learning.