ABSTRACT

Many students at German universities have difficulties to cope with the introductory courses on theoretical computer science, which results in very high dropout and failure rates. In the past years we have experimented with a variety of improvements to the theory course held at the University of Potsdam, Germany. During last winter semester we improved the setup of the course by taking into account the pedagogical approach of cognitive apprenticeship. The result of adopting this approach was a reduction of the failure rate to only 6%. In this paper we describe and discuss the adjustments to the setup of our course. For the next winter semester we plan to repeat the course in the same way and to evaluate its quality.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: Computers and Education - Computer and Information Science Education

General Terms
Human Factors

Keywords

1. INTRODUCTION

The department of Computer Science (CS) at the University of Potsdam, Germany, offers undergraduate and graduate programs in CS. The course “Introduction to theoretical computer science I” (briefly “Theory I”) is a mandatory course for all undergraduate CS majors. Since 2003 it has been taught annually by the second author. The first author co-taught it in the winter of 2011/12. The course usually has high dropout rates, since many students have problems understanding and following the course topics. A learning edge momentum [13] can be observed very early in the course as well. Our tutors have observed that students had already major difficulties with mathematical notations and formalisms, which prevented them from dealing with the true theoretical CS content. Also the nature of wrongly solved assignments in the final examinations indicated a lack in mathematical background. To solve this problem, we offered a two weeks pre-course in mathematical basics and also covered mathematical proofs during the first two weeks of the course and practiced them in the first weekly assignments. Despite this additional offer, students’ problems to cope with the lecture's topics remained the same, which forced us to reconsider all elements of the course. Similar problems in equivalent introductory courses to theoretical CS were reported by Chesñevar et al [3] who introduced several improvements based on constructivism. Hämäläinen [6] used problem-based learning and Korte et al. [7] developed a constructionist approach with game-building. All approaches succeeded to engage students in more learning activities resulting in lower dropout and failure rates. In addition, Rodger et al. [14] suggested the visualization tool JFLAP for formal languages and automata theory, which shall enable students to interact more with the introduced theoretical concepts.

In the past years we have experimented with a variety of improvements to our Theory 1 course. During last winter semester we improved the setup of the course by consolidating our hands-on experiences and taking into account the situated cognition theory by Brown et al. [2] and the pedagogical approach of cognitive apprenticeship by Collins et al. [5]. These authors emphasize that formal schooling focuses mostly on knowledge transfer while neglecting the situated activities and contexts in which learning takes place. In addition, knowledge and skills required for the process of handling knowledge remains mostly tacit and is invisible for the students ([2], p. 32). Cognitive apprenticeship is well known in education and has already been applied in CS education. In the literature, however, there is no report that it has been used in a theoretical CS course.

The result of adopting cognitive apprenticeship was that only 6% of students attending the course failed the final exam, which encouraged us to continue these changes and to attempt an empirical evaluation of our pedagogical approach. In this paper, we will introduce cognitive apprenticeship and describe the adjustments to the setup of our course based on this approach.
2. **THE THEORY I COURSE**

There are a few differences between the German and the American system of higher education that are worth mentioning before going into the specifics of the Theory I course. German students enter an undergraduate program after finishing 13 years of school and immediately focus on their major. Many of the mandatory courses for CS students require a good understanding of mathematical and theoretical foundations. Since there is no tuition and no selection process, students often choose a CS major based on a fascination for technology, programming, or computer games but without an understanding what computer science is about. As a result, CS dropout rates in Germany have averaged about 50% since the 1970s. Most students give up during the first two semesters. The German system has a winter and a summer semester with teaching periods between mid-October and mid-February and between mid-April and mid-July, respectively.

2.1 **Course Content**

The “Introduction to theoretical computer science” at the University of Potsdam is a two-semester course on the foundations of automata, programming languages, computation, and computational complexity. It introduces idealized mathematical models and discusses methods for designing and analyzing them. The Theory I course focuses on the theory of automata and formal languages. It begins with an investigation of simple automata models and increases the level of complexity until the models reach the capabilities of modern computers. Specifically it covers the following topics.

- Regular languages: finite automata, nondeterminism, regular expressions, type 3 grammars, closure properties, and limitations
- Context free languages and type 2 grammars, pushdown automata, normal forms, parsing algorithms, closure properties, and limitations
- General and context sensitive languages and grammars, Turing machines, linear-bounded automata, closure properties

In the course of the lectures, students are supposed to develop specific skills that enable them to work with the models introduced in the course. For this reason, specific skills that describe the intended learning outcomes were normatively defined ([10], p. 36-38). For example:

- **Subject specific skills**: students are able to analyze deterministic and non-deterministic automata and grammars with mathematical methods and to prove their correctness; students know methods for translating between different automata model, grammars, and regular expressions and are able to justify the correctness of these conversions.

- **Methods skills**: students can employ mathematical proof techniques for the analysis of automata and formal languages; they can transform grammars into normal forms; they are able to prove whether a given formal language is regular (context free) or not.

- **Behavioral skills**: students are capable of working in small teams when developing solutions for given problems; they are able to give an oral presentation of their solutions and they can write them down in a precise language.

The number of students attending the course usually varies between 150-300 and consists of 50-150 students majoring in CS, 100 students majoring in Software-Engineering, 40 students majoring in business informatics, approx. 10-25 prospective teachers with one of their majors in CS, and approx. ten students majoring in different fields (like linguistics or others).

CS and business informatics majors are supposed to attend the course in their first semester and software engineering majors in their third. In many other German universities, CS majors attend a course in theoretical CS in their third or fourth semester. At the University of Potsdam the course was explicitly set up at the beginning of CS undergraduate studies, since many subsequent courses expect students to be familiar with automata theory and Turing machines.

Because of the large number of participating students, the course assessment is accomplished with one final written exam at the end of the course. The failure rate usually ranges between 30% and 50%. A considerable amount of students drops out of the course or even the whole CS major before the final exam. Since the course is held during the winter, most students who drop out do not return after the Christmas break. Currently, the dropout rates for the CS major in Potsdam are below the national average, but for the course they are still too high.

2.2 **Course Setup**

Until we modified the course setup and its pedagogical approach, the course consisted of the following components, which are typical for an introductory CS course offered in Germany:

- Three hours of lectures peer week by a faculty member who presented the course topics, central concepts, algorithms, and their proofs and illustrated them with examples.

- Weekly homework assignments, which the students had to solve individually and submit in writing for grading. One hour exercise sections per week held by tutors (usually senior students), during which solutions to the assignments and some of the lecture's topics where discussed in depth.

During exercise sessions, it was common to have one of the students present his or her solution to everybody else. The tutor chaired the session and took care that the solution was correct and helped in case of problems. It was assumed that students understood the presented concepts, theorems, and proofs during the lecture and were able to deal with the weekly assignments by themselves. The weekly exercise sessions were supposed to help the students check the correctness of their own solutions. However, most tutors reported that students remained very passive during exercise sessions and did not participate in discussions even when their own solution contained mistakes. Because of the high failure rate in the finals we concluded that most of our students were not able to understand the contents of a lecture solely by listening to it. It also seemed obvious that the course setup made implicit assumptions about capabilities that a majority of students did not yet have.
3. CONCEPTUAL FRAMEWORK

How do people learn and under which circumstances do they perform well? There are several theoretical approaches that attempt to give an answer to this question. In this section, we will discuss situated cognition theory, contrast it with cognitive understanding, and discuss its impact on formal schooling.

3.1 Situated Cognition Theory

Cognitivist approaches were influential in the second half of the last century and are probably the most dominating perspective on learning in CS Education. The cognitive view focuses on the question how knowledge is acquired and represented in the mind. Different forms of knowledge are distinguished like general or domain-specific knowledge, which includes “concepts, facts, and procedures explicitly identified with a particular subject matter” ([4], p. 49), as well as declarative (“knowing that”), procedural (“knowing how”), and conditional knowledge (“knowing when and why” to apply declarative and procedural knowledge) ([16], p. 258). Learning is conceptualized as a process in which students create a mental model of a specific knowledge entity in their minds. A person’s cognitive processes operate on such mental models, based on logic-like rules of inference, and are understood to happen solely in the person’s mind. From this point of view successful learning is, roughly speaking, based on the teacher’s ability to present or expose domain knowledge as well as the students’ cognitive ability to follow the teacher’s demonstration in order to develop a mental model that represents what the teacher was demonstrating ([16], p. 248ff).

In the last thirty years many studies provided new insights into what we know about learning; see for example [8], [15], and [11]. A variety of approaches with a very different understanding of learning emerged from this research, such as activity theory, situated learning, socio-cultural theory, social constructivism, or situated cognition theory [12]. Because of space limitations we will only summarize the key characteristics of this family of approaches, focusing on situated cognition theory introduced by Brown et al. [2].

In contrast to cognitivism the approaches have in common that cognitive processes are viewed as inseparable from a person’s activities, the socially and culturally shaped environment in which activities happen, and the tools (physical or conceptual) the person uses to accomplish them. Domain knowledge, no matter if of declarative, procedural, or conditional nature, is understood to be always contextualized within the situation in which it was developed, representing specific meaning that is “inherited from the context of use” ([2], p. 33). Brown et al. understand domain knowledge as a specific conceptual or mental tool and argue that learning how to use knowledge as a tool “involves far more than can be accounted for in any set of explicit rules. The occasions and conditions for use arise directly out of the context of activities of each community that uses the tool, framed by the way members of that community see the world. The community and its viewpoint, quite as much as the tool itself, determine how a tool is used. […] Just as carpenters and cabinet makers use chisels differently, so physicists and engineers use mathematical formulae differently.” ([2], p. 33)

From the situated cognition perspective students succeed when their learning process is situated, which means that the situation in which domain knowledge is taught corresponds to the situation in which it was developed and is still used. Collins et al. [5] draw on research investigations by Lave [8] and argue that for the longest time in human history the natural form of learning was apprenticeship: a master-students relationship that focuses on practicing contextualized knowledge and skills in authentic situations that provide meaning to the activities involved. Learning was not only a form of “acquiring” knowledge and developing skills to handle it but also a form of enculturation into a certain community. Being set in a specific workplace, tasks and problems arose not from pedagogical concerns but from the demands of the authentic environment ([4], p. 48ff). Of this master-student-relationship Collins et al. describe the following techniques to be essential:

- **Modeling** where the teacher demonstrates how to do something and makes single steps of a process visible such that students can observe and be than able to imitate it.
- **Coaching** where the teacher guides or supervises students’ activities, efforts, and experiences evaluating them, offering encouragement and feedback.
- **Scaffolding** is the framework or certain plan the teacher provides student with in order to carry out specific tasks.

Nowadays, master-student-relationships can be found for example in learning to play an instrument or to do research as well as multiple forms of trades like plumbing or tailoring.

3.2 Formal Schooling and Cognitive Apprenticeship

With regard to situated cognition theory, Collins et al. argue that typical forms of academic formal schooling are the opposite of apprenticeship and reflect mostly a cognitive view of learning ([5], p. 38-39), ([4], p. 47ff). In a lecture the teacher introduces and demonstrates declarative and procedural knowledge as a stand-alone generalized product, which is decontextualized from the situated activities in which it was once developed and in which it is supposed to be used by the students in future. As a consequence, conditional knowledge remains tacit since a specific context is not emphasized. But for the members of the scientific research community who created this knowledge, it is not a decontextualized stand-alone product, but a cognitive tool they use and apply in activities where it is meaningful and relevant to them. Bereiter ([1], p. 295ff) argues that learning in formal schooling is particularly difficult for two reasons. First, there is no sufficient distinction between knowledge as the material or item of inquiry and knowledge and skills that are needed to handle this material; second, the latter remains mostly tacit since little attention is paid “to the reasoning and strategies that experts employ when they acquire knowledge or put it to work to solve complex or real-life tasks. […] To make real differences in students’ skill, we need both to understand the nature of expert practice and to devise methods that are appropriate to learning that practice” ([5], p. 38-39).

Collins et al. argue that in contrast to craft, activities that handle “knowledge material” are in part invisible and that therefore it is almost impossible for students to observe and imitate them in the same way as in traditional apprenticeship. Still, students in higher education institutions are supposed to adopt the expertise of a particular scientific community and to be able to work with the domain knowledge introduced in a lecture. As a pedagogical
approach that focuses on the gap between process and product. Collins et al. suggest cognitive apprenticeship, [4], [5]. The idea of this approach is to focus not just on domain knowledge as a decontextualized product but to make the required skills and knowledge more explicit and the thought processes of teachers and students more visible ([5], p. 40). For this reason, they propose to use the key methods of traditional apprenticeship as well as additional methods specific to their approach ([4], p. 50ff): The teacher is supposed to explain and demonstrate knowledge and skills (modeling and articulation), which students can observe and then repeat under the guidance of the teacher (coaching). Different frameworks are proposed to the students that help them to orientate themselves and their learning activities (scaffolding). Since most of the activities are cognitive and not visible, it is important that teachers and students develop the ability to articulate and reflect their activities.

Another important aspect of cognitive apprenticeship is providing the context in which domain knowledge is meaningful. In the workspace of traditional apprenticeship, reasons for specific activities are much better understood than in formal schooling: students "are motivated to work and to learn the subcomponents of the task, because […] they have seen the expert's model of the finished product, and so the subcomponents of the task make sense. But in school, teachers are working with a curriculum centered around reading, writing, science, math, history, etc. that is, in large part, divorced from what students and most adults do in their lives. In cognitive apprenticeship, then the challenge is to situate the abstract tasks of the school curriculum in contexts that make sense to students” ([4], p. 50).

4. ADJUSTMENTS TO THE COURSE

Situated cognition theory helped us understand that with a course setup as described in section 2.2 the emphasis lies mostly on presenting theoretical CS domain knowledge (the concepts, definitions, algorithms, etc.) as stand-alone products and does not sufficiently explicate the methods, approaches, and strategies of dealing with this knowledge as well as techniques how to learn them. As a consequence, activities (see 2.1) that students are supposed to perform in order to handle theoretical CS knowledge remain tacit and are not addressed and exposed sufficiently.

We are aware that teacher-to-learner ratio in traditional apprenticeship is very small. Thus the approach is not directly applicable in modern schooling and surely impossible in a course attended by hundreds of students. Since we did not have access to additional resources, we were wondering to what extent we could implement what we know about learning with the resources we had. It was not our intention to completely change the course setup, which appeared to have worked for many years and many students, but to improve the course setup with regard to cognitive apprenticeship taking into account the skills students have to develop in order to cope with the course topics.

4.1 Lecture and Tutorial

The goal of the lecture is to present new topics, central concepts, algorithms, and their proofs and to illustrate them by examples. So, the lecture corresponds roughly to the method of modeling. But since all lecture topics are prepared in advance for a smooth presentation, students experience them as knowledge products without seeing the enormous effort it took to create them. Skills to handle and work with this knowledge that students are supposed to develop in the course are only implicitly demonstrated during the lecture. Also the lecture lacks interaction between students and the instructor, since there is little time for individual questions and discussions.

The first change we implemented was to shorten the lecture to two hours and expand the exercise session to two hours per week in order to increase student activities. For this reason, lecture topics are now presented on slides that are prepared in advance instead of being written onto the blackboard. This enables the teacher to focus on explaining instead of writing and to present the same amount of topics in less time. In order to offer students the possibility to prepare before and after the lecture, the slides were made available at the beginning of the entire course. Furthermore the lectures were recorded on video and posted on the course webpages a few days later.

Next, we installed another weekly two hour session that we named tutorial. In the tutorial, which is held by the instructor, students are encouraged to ask questions and discuss issues that were still unclear after lecture and exercise sessions. The topics discussed in the tutorial are not prepared in advanced but suggested by the attending students, as the main objective is not the solution to a problem but the process of producing it. When students experience that even the professor has to try different approaches and alternatives before accomplishing a proof, they will learn that this is the normal procedure of dealing with assignments. While the lecture presents the ready-made “products” of theoretical CS, the tutorial is one place in the course where the process of creating them is demonstrated with regard to modeling.

4.2 Exercise Session and Assignments

We redesigned the weekly exercise sessions and assignments entirely with regard to the coaching and scaffolding method.

Now, the session and the assignments contain three parts:

- **Quizzes** contain approximately five right-or-wrong statements related to the topics of the previous week’s lecture. Answers are discussed in the first 15 minutes of the session. This helps students to check their understanding and serves as a warm-up and first discussion.
- **Exercises** are solved jointly during the exercise session and serve as preparation for homework. Written solutions to the exercises are posted on the webpages after the sessions.
- **Homework assignments** are to be solved by teams of 2-4 students without support from tutors or instructors and to be submitted for grading in written form.

Students are supposed to attend the lecture and one of the weekly exercise sessions and to form study groups of 2-4 people that work regularly on the homework assignments. The course is organized as follows: The lecture takes place on Friday morning. On the same day a new quiz, exercises, and homework for the next week are published on the course’s webpage. Over the weekend before students have time to study both quiz and exercises. The exercise sessions take place from Monday till Wednesday and the solutions to the exercises are published immediately after the last exercise session. Afterwards, students
are expected supposed to work with their study groups on the homework assignments, which are due the following Monday.

The tutors responsible for the weekly exercise sessions shall encourage the students to participate actively in ongoing discussions. The main idea is to use the exercise session as a form of coaching, to enable students to solve homework by themselves, and to motivate them to become more actively involved in the exercise sessions. In order to accomplish this, we aligned the exercises with the homework with respect to structure and content. This means that in each session the same type of problems is used for the exercises as well as the homework assignments.

The solutions to the exercises demonstrate what a correct solution should look like in written form and what is expected from the students when they turn in their homework. Especially for the latter there is not enough time during the exercise session, since possible solutions to the exercises are developed together and only sketched on the blackboard. But students must also learn how to produce a complete and precisely formulated solution using mathematical formalisms. The alignment between exercises and homework together with the solutions to the exercises is the scaffolding framework for students.

4.3 Submission of Homework and Final Exam
As prerequisite for the final exam, students must submit weekly homework assignments. Each submission is graded and commented by the tutors. It is very important to make submissions mandatory. Otherwise students do not work regularly enough to succeed in the final exam. Students must obtain at least 50% of the possible points to be admitted. More than 90% of all students manage to fulfill this requirement.

The weekly homework grading as well as the tutors’ support during the exercise session is a form of coaching. In addition, it corresponds to students’ activities during final exam, where students have to formulate a written solution for an assignment, which then will be graded by the tutors under supervision of the instructors. In order to understand the expectations, especially with respect to the very strict and formal character of solutions to assignments, students need to train this skill as well and to receive a weekly feedback about their efforts.

The final exam is written one week after the end of the lectures during the assessment period, which in the winter takes place at the end of February. Shortly before the two weeks of Christmas break we also assess students by a pre-exam, which counts as one submitted homework. Since the assessment is a situated activity, students must experience and learn how to deal with this as well. The pre-exam has the same form and amount of assignments as the final exam and gives the students an opportunity to practice the assessment situation. In addition, the pre-exam shows students what will be expected from them during the final and the results give them another direct feedback about their current effort and achievements.

The final exam contains 7-12 Quiz questions and 6-8 assignments with 2-4 sub-assignments each, which are aligned with the homework assignments students are already familiar with. Each sub-assignment in the exam assesses one particular skill without relying on results from previous sub-assignments. In addition, we used typical automata, regular languages or Turing machines with a simple structure and without containing unusual cases. We did this on purpose: our goal was to assess the students’ skills to handle these artifacts and we wanted to test if students are able to apply appropriate transformations and proof techniques to them. We believe that using irregular or complicated examples during a final exam can distract students and cause them to spend too much time on figuring out the details of the specific objects instead of dealing with the assignment itself.

4.4 Discussion
In the first weeks of the course we repeatedly explained the course concept and its different course elements during the lecture, the tutorial, the exercise sessions, and as written information together with the first weekly assignments. Before the pre-exam and final exam we also explained precisely how both exams are organized, passing suggestions how they should prepare themselves.

Most students who attended the course during the winter term 2011/2012 appeared regularly during exercise sessions and participated actively during the quiz and working with the exercises. The tutorial was attended by many more students than in the years before, when we did not align homework assignments with the exercises. During the tutorial, students asked very precise and concrete questions related to the contents of the lectures and the exercises. They also asked for solutions to the more difficult assignments already submitted. In the years before, students mostly asked questions about the current homework assignments, which were difficult to answer without providing solutions to tasks still ahead of them. With the alignment between homework and exercises, students seemed to be more successful in solving homework assignments by themselves once they had attended the exercise sessions, seen a written solution to the exercises, and discussed the more difficult aspects in the tutorial. Altogether, our impression was that our changes helped students to be more focused, to stay motivated, and to keep working on their weekly assignments.

We are aware that due to our limited resources the changes to the course had to be small and simple. We were not yet able to accomplish all aspects of cognitive apprenticeship. For example, it is important that students articulate their activities and reflect on them ([4], p. 51). Since we cannot change the teacher-learner-ratio and introduce more study sessions, most of students’ activities can be only scaffold. The rest has to happen during their individual studies. We only have little influence on the latter and know that when students manage to find an adequate study group they perform very well. But a lot of students are not able to do this and besides encouraging them to look for a study group, we have not found a method how to support this better.

Furthermore, it remains open if and how to situate ([4], p. 52ff), the course topics and abstract tasks of the weekly assignments in contexts that make sense to the students. For theoretical computer scientists, the context of their work remains the theoretical field with its abstract problems. The meaning of their activities comes from the intellectual challenge and interest for working with these kinds of knowledge objects. But most students have no intention to become part of this specific scientific community. Nevertheless, in its role as a cognitive tool the domain knowledge of theoretical CS will become meaningful for them as they rely on it during their whole professional career.
Addressing the same issue of contextualization, Chesñevar et al [3] introduced “biographical notes, videos and articles associated with the historical context in which the theory of computing emerged as a new discipline” (p. 8). The authors report that students responded very well to this historical perspective of theoretical CS, developed a deeper understanding, and became therefore more engaged. This is an interesting observation since for the scientific community the historical context is not what makes their interaction with theoretical CS meaningful. But the historical context helped students understand the meaning of the course’s topics by studying what they meant to the community in the past. Nevertheless, it is not yet clear if understanding the historical context will actually reveal why the topics should be meaningful for a student’s future activities. It is surely important and worth to be taken seriously, but we believe that it is more important to emphasize the course’s topics role as a cognitive tool and how it can be used and applied, which provides meaning as well. Altogether, it is important for us to elaborate on this issue in future and to emphasize it in the Theory 1 course in a way that goes beyond the traditional teaching approach of providing short examples of a topic’s possible applicability.

5. CONCLUSION

In this paper we introduced a pedagogical approach to teaching the introductory course on theoretical computer science at the University of Potsdam that changed the course setup based on insights from situated cognition theory and cognitive apprenticeship. Our goal was to realign the course with respect to the historical context in which the theory of computing emerged as a new discipline and to make the practices of theoretical CS more visible. We reinforced modeling by introducing a tutorial session; we scaffold students’ activities through a strong alignment between exercises, homework, and the final exam; and we tried to coach students’ activities in the exercise sessions and by providing feedback for their homework submissions. Students seem to respond very well to this learning environment. Only 6% failed the final exam.

Taking the final exam’s result as a first indicator, we plan a more thorough evaluation of our approach in the coming winter semester using a mixture of quantitative and qualitative research approaches. We also want to assess the impact of the different elements of the course with regard to students’ background and preknowledge. Specifically, we want to assess if the alignment between exercises and homework serves as the intended scaffolding framework and enables students to work in a way that prepares them for the final exam. For this purpose, we plan a survey study with all students at the beginning and the end of the course as well as in the week before the Christmas break. In a qualitative study, we plan to interview a small number of students about their weekly study sessions. Furthermore, it would be important to conduct a study that observes how students solve their weekly homework and work together in their study group. In order to validate our data it will be also important to conduct a similar evaluation at another German University that also offers a Theory 1 course in the first semester, but uses the traditional teaching approach. Such a comparison would provide further insights if our students know better how to solve their assignments and to prepare appropriately for their final exam.

REFERENCES
