Education Tech Research Dev (2014) 62:555–570
DOI 10.1007/s11423-014-9345-6

DEVELOPMENT ARTICLE

GearSketch: an adaptive drawing-based learning environment for the gears domain

Frank A. J. Leenaars · Wouter R. van Joolingen · Hannie Gijlers · Lars Bollen

Published online: 25 July 2014

Abstract GearSketch is a learning environment for the gears domain, aimed at students in the final years of primary school. It is designed for use with a touchscreen device and is based on ideas from drawing-based learning and research on cognitive tutors. At the heart of GearSketch is a domain model that is used to transform learners' strokes into gears and chains, animate the turning of the gears and check whether learners' solutions to practice problems satisfy the given constraints. Additionally, this domain model is the basis for GearSketch's learner model and item generation an selection mechanisms. The learner model is used to track learners' knowledge and adaptively select items as they progress through the practice problems. Two experimental evaluation studies show that Gear-Sketch's interface and animations lead to improved learning outcomes, but that its adaptive features do not significantly affect posttest results.

Keywords Drawing · Simulation · Problem solving · Cognitive tutors · Gears

Introduction

Ever since computers found their way into education, they have been used to support learning how to solve problems. Among the famous examples of problem-solving tutors

F. A. J. Leenaars (✉) · W. R. van Joolingen · H. Gijlers · L. Bollen
University of Twente (GW/IST), PO Box 217, 7500 AE Enschede, The Netherlands
e-mail: f.a.j.leenaars@utwente.nl

W. R. van Joolingen
e-mail: w.r.vanjoolingen@utwente.nl

H. Gijlers
e-mail: a.h.gijlers@utwente.nl

L. Bollen
e-mail: l.bollen@utwente.nl

W. R. van Joolingen
Freudenthal Institute, Utrecht University, Utrecht, The Netherlands

are cognitive tutors based on ACT[1] theory, such as the LISP[2] tutor, and algebra and geometry tutors (Anderson et al. 1995), as well as Andes, for learning to solve physics problems (VanLehn et al. 2005). Such tutors have in common their modeling of a domain with a set of well-defined rules, such as geometry proofs, in terms of declarative and procedural knowledge. These cognitive tutors have been quite successful in increasing performance on knowledge tests. In a different strategy, called constraint-based tutoring (Mitrovic et al. 2001; Mitrovic 2003), the focus is not on arriving at one single, perfect solution to a problem, but on fulfilling the domain's constraints, which may lead to alternative, equally correct solutions. Here, the domain description, learner model and feedback mechanism are based on constraint matching, error recognition and error correction. By modeling both the domain and the learner, cognitive tutors can adaptively select practice problems for each individual learner (Mitrovic and Martin 2004). Instead of offering the same static sequence of problems to all students, the cognitive tutor can select appropriately challenging problems for each learner individually, based on its model of their current knowledge. Mitrovic and Martin (2004) found that this can have a positive effect on learning performance, especially for students who are below or above average ability.

Another approach to supporting complex problem-solving that until recently fell outside the scope of computer-based tutoring is the creation of drawings. Creating a drawing helps learners self-explain problems while working on them (Cox 1999), can help when solving arithmetic word problems (Van Essen and Hamaker 1990), and supports scientific reasoning (Ainsworth et al. 2011) and the modeling process (Leenaars et al. 2013). The increasing availability of pen-based and touchscreen devices in (primary) schools (Hu 2011; Lee 2010), makes it possible to combine ideas from technology-enhanced learning and drawing-based learning. Two examples of digital learning environments that interpret students' drawings are CogSketch (Forbus et al. 2011) and SimSketch (Bollen and Van Joolingen, 2013). It also allows new learning environments to combine ideas from cognitive tutors and learning by drawing. Such an environment adopts from drawing-based learning the idea of offering an expressive interface in which students can easily represent the systems they are reasoning about by making a quick sketch. From cognitive tutoring, this environment adopts modeling individual learners' knowledge of the domain and using these learner models to adapt the tutoring process to each individual learner.

Such a learning environment is well suited for supporting learning to solve complex problems in a domain with well-defined rules and a natural graphical representation. The gears domain has all three. Furthermore, because young learners are familiar with gears from everyday experience (e.g. bicycles, wind-up toys, clocks), gears are suitable as reference objects during the introduction of abstract concepts in mathematics and physics (Bartolini Bussi et al. 1999. In previous studies, gears have been used to introduce parity (Dixon and Bangert 2004), fractions (Andrade 2009) and mechanical advantage (Chambers et al. 2008). Although the rules governing the gears domain are simple, it is easy to create gear configurations in which gears interact in complex ways. Additionally, when the goal for students is to arrange gears in such a way that given constraints are satisfied, as well as to predict how a gear configuration will behave, the task becomes even more challenging. Besides having to know the rules governing interactions between gears, students must now also recognize which rules are potentially useful and figure out how to deal with the spatial

[1] Adaptive Character of Thought.

[2] "List Processing", a family of programming languages.

constraints inherent in any concrete configuration of gears. Taken together, these challenges involved in creating a gear configuration that satisfies certain constraints constitute a complex problem that does not have a single, exact solution.

In this paper we discuss the development of GearSketch, an adaptive drawing-based learning environment for the gears domain, aimed at primary school students and designed to be used with a pen-based touchscreen. The development questions addressed are:

- DQ1: What kind of problems should students learn to solve by working with GearSketch and how can the required domain knowledge be modeled?
- DQ2: How can ideas from drawing-based problem solving be used to design a simple interface for practicing with these problems?
- DQ3: How can ideas from cognitive tutors be used to design a learner model that tracks students' knowledge as they progress through practice problems?
- DQ4: How can suitable practice problems be automatically generated and selected?

Answering these development questions is the main goal of this paper. Additionally, the following evaluation questions are addressed by a brief discussion of one previously reported and one new experimental study:

- EQ1: Does an interface that interprets and animates students' drawings help students learn more from working with GearSketch?
- EQ2: Does tracking individual students' knowledge with a learner model and using this model to select appropriate practice problems help students learn more from working with GearSketch?
- EQ3: Can this learner model make accurate predictions about students' performance on a posttest?

The next section introduces GearSketch and addresses the development questions. The section after that discusses our experimental evaluation of GearSketch and addresses the evaluation questions.

GearSketch

GearSketch is software that supports students in the final years of primary school in learning to solve problems in the gears domain. When learners work with GearSketch, they begin by working through a series of integrated tutorials that explain the relevant domain theory and teach them to work with the software. For example, when students are learning about the turning directions of gears connected via chains, the tutorial first explains that gears on the same side of a chain will turn in the same direction, while gears on opposite sides will turn in opposite directions. Next, students draw a chain that connects gears to one another. Then they are asked to predict the directions in which the gears connected by the chain will turn, if one of them is turning in a given direction. Finally, they can check whether their predictions were correct by watching an animation of the gears turning. After finishing the tutorial, students apply and strengthen their domain knowledge by answering questions and solving puzzles.

The next two sections discuss GearSketch's domain model and how this is used by the interface. Two subsequent sections discuss GearSketch's learner model and item generation features.

Domain model

The domain of gears and chains is governed by rules that together make up GearSketch's domain model. These rules define the interaction between gears and chains in two ways. First, they define the relative speeds with which connected gears turn given their size ratio and connection type. Qualitative versions of these rules are listed in Tables 1 and 2. Second, they define constraints on possible spatial layouts. For instance, two meshing gears may not also be connected by a chain, because then they would be unable to turn.

Table 1 summarizes how gears transmit motion for four different connection types. In this table "turning speed" refers to the angular velocity of the gear and "tooth speed" refers to the linear velocity of the gear's teeth. For each connection type, the connected gears will have either equal turning speed or equal tooth speed. When two connected gears' equal speed type is known and the relative size of the gears is also known, conclusions can be drawn about the other type of speed for the two gears. Table 2 summarizes the relevant rules. Students who can reproduce the rules in Tables 1 and 2 from memory have declarative knowledge of these rules. However, using this information to solve problems requires procedural knowledge, in the form of production rules (Anderson et al. 2004). For each declarative rule, multiple production rules can be created. For instance, two production rules for the turning directions of meshing gears are:

1. IF gear A is turning clockwise and gears A and B mesh.
   THEN gear B will turn counterclockwise.
2. IF gear A is turning clockwise and gear B should turn counterclockwise.
   THEN this can be achieved by connecting gears A and B so that they mesh.

The first of these rules is the type used to explain the behavior of a gear configuration and to predict what will happen if changes are made. The second rule is the type used to create configurations that satisfy given constraints. Figure 1 shows a question item that can be solved by applying the first type of production rule, which follows quite directly from the declarative rules. The steps a student must take to answer this question can be summarized as follows, where TuD(X) means "turning direction of X":

1. TuD(B) = clockwise	[given]
2. TuD(B) $\neq$ TuD(C)	[turning direction of gears on opposite sides of a chain]
3. TuD(C) = counterclockwise	[follows from step 1 and 2]
4. TuD(C) = TuD(D)	[turning direction of gears connected via their axes]
5. TuD(D) = counterclockwise	[follows from step 3 and 4]

If the student knows the two relevant declarative rules and applies them correctly, finding the answer is straightforward. This is not the case for puzzle items, where recognizing which rules are relevant is more difficult and insight into the spatial properties of the task is necessary to find a solution. To solve the puzzle shown in Fig. 2, students need to know the same two rules required for answering the question in Fig. 1. Additionally, they need to recognize that these are the relevant rules for solving this problem, place gears B and C in correct locations and correctly connect the gears with a chain. GearSketch needs to offer students who are learning to solve puzzles like this a simple way to represent and examine multiple possible solutions. To achieve this, GearSketch has a computational model of the domain rules as summarized in Tables 1

Table 1 Rules relating connection type with turning direction and speed

Connection type	Turning direction	Equal speed
Meshing gears	Opposite	Tooth speed
Gears on top of one another (connected via their axes)	Equal	Turning speed
Gears on the same side of a chain	Equal	Tooth speed
Gears on opposite sides of a chain	Opposite	Tooth speed

Table 2 Rules relating gear size, turning speed and tooth speed

Relative sizes	TuS(X)[a] = TuS(Y)	ToS(X)[b] = ToS(Y)
Size(X)[c] = size(Y)	ToS(X) = ToS(Y)	TuS(X) = TuS(Y)
Size(X) > size(Y)	ToS(X) > ToS(Y)	TuS(X) < TuS(Y)

[a] The turning speed of gear X

[b] The tooth speed of gear X

[c] The size of gear X

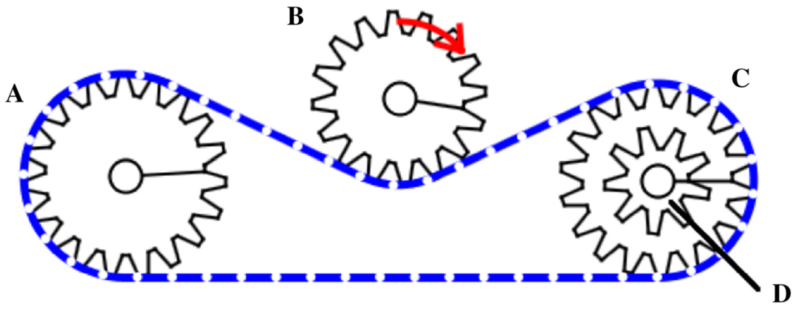


Fig. 1 Example question: "Will gear D turn clockwise or counterclockwise?"

and 2, as well as of the ways in which gears can be configured spatially. This computational domain model is the basis for GearSketch's drawing-based interface that interprets and animates learners' drawings.

Interface

Figure 3 shows a screenshot of the GearSketch interface. In the upper left corner there are five large buttons for the different tools the learner can use to add and remove annotations, gears, chains and turning arrows. The sixth button is the play button, which starts the simulation. The three buttons in the upper right corner are used to reset a puzzle to its original state, check the learner's solution and go to the help menu, in order from left to right. The help menu gives an overview of the declarative rules in Tables 1 and 2. The large area in the middle is for drawing the gears and chains, and the blue box at the bottom of the screen can contain questions or instructions for the puzzles.

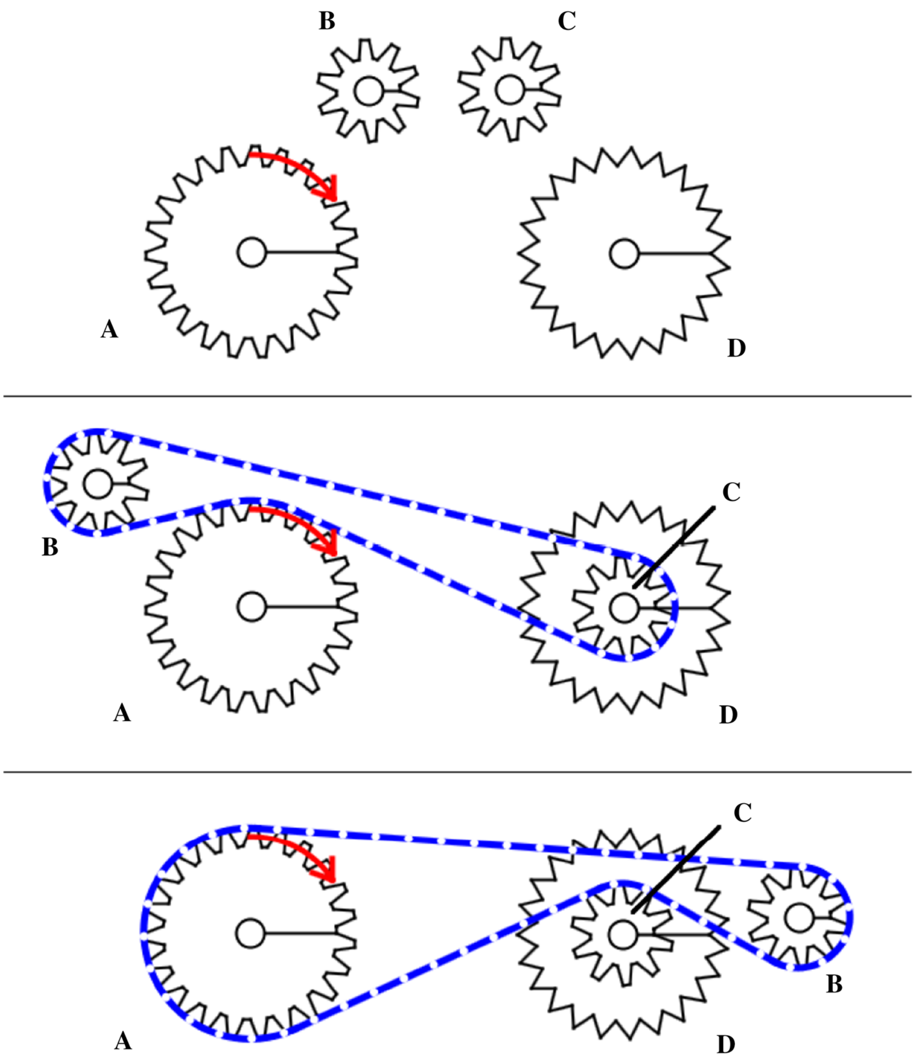


Fig. 2 Example puzzle: "Add a chain so that gear *D* will turn counterclockwise. Only gear *B* and *C* can be moved. Gear *D*'s sharp teeth mean that they cannot be connected to a chain or another gear's teeth." Two possible solutions are shown

The interface is designed to be intuitive to use with a pen-based touchscreen. Gears are added by simply drawing a circle while the gears button is selected. A gear the same size as the circle will appear in its place when the stylus is lifted from the screen. Gears can be removed by crossing them out and be moved by dragging them. When a gear is dragged close to another gear it will snap to connect to the other gear, with its teeth aligned correctly. This also works when a gear is used to connect two other gears to each other. A smaller gear can be dragged on top of a larger gear and will then snap to connect to its axis. This means that gears can overlap when they are on different levels, as can be seen in the configuration on the left side of Fig. 3. Because the gears are slightly transparent, underlying gears can still be seen.

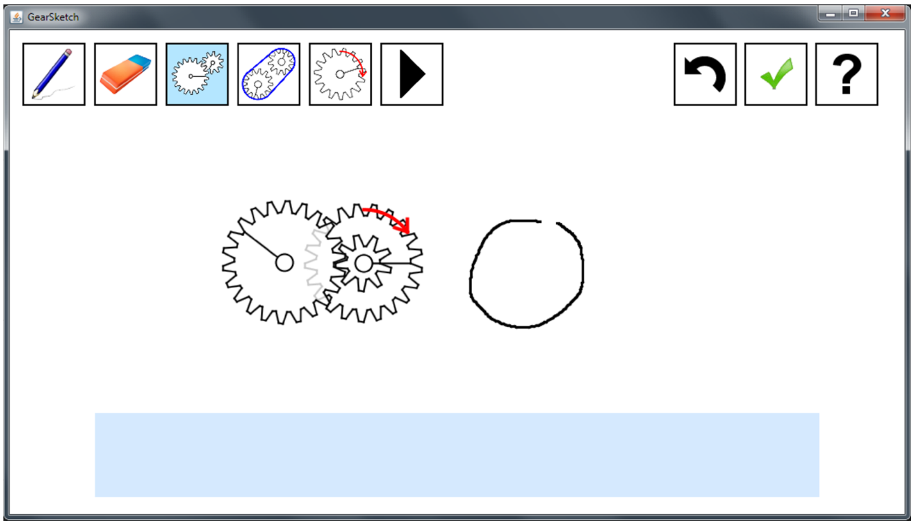


Fig. 3 Screenshot of the GearSketch interface showing a new gear being drawn

A chain is added by selecting the chains tool and drawing its outline. GearSketch will automatically tighten the chain around its supporting gears. Automatically tightening the outline of a chain is a problem of finding the shortest homotopic path (Bespamyatnikh 2003), with some added complexity due to the different levels of gears. When a chain is drawn through a gear, the chain could be invalid, but the learner could also intend the chain to be on a different level than the gear it is drawn through. Figure 4 shows how GearSketch interprets and tightens a chain in a situation with multiple levels of gears. Once a chain has been added, it can be removed by crossing it out. When supporting gears are moved or other gears are moved into the chain, the chain will react elastically, growing and shrinking as necessary.

When learners are moving and adding gears and chains, GearSketch uses its knowledge of the spatial constraints in the domain model to ensure that no invalid configurations are created. A configuration is considered invalid when its gears cannot turn, such as when three meshing gears are connected circularly or when two meshing gears are connected by a chain. GearSketch also makes sure that gears and chains that are on the same level do not overlap, while gears and chains on different levels can. The level objects are on is determined by analyzing the connections between them. The complexity of the algorithms that ensure the validity of configurations created by learners is directly related to the complexity in the domain model. The rules that determine the validity of a given gear configuration are not found in Tables 1 or 2, but are also part of the domain model; learners need to know these rules to create their own gear and chain systems to solve puzzles in this domain.

When the play button is selected, GearSketch uses its knowledge of the domain model and of how the gears are interconnected to animate the model. Gears with a red arrow in them, which learners can add with the arrow tool, will start to turn. All gears and chains connected to these moving gears will also simultaneously start turning, according to the rules of the domain model. This allows the learner to explore the behavior of the current gear configuration and to see what happens when changes are made. The first time learners attempt to answer a question or solve a puzzle, the play button will be disabled. This means that they must reason about the behavior of the system to predict what will happen rather

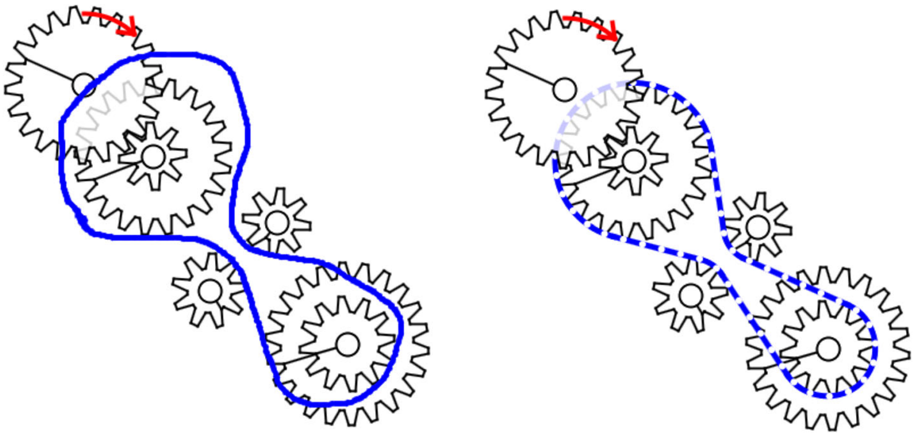


Fig. 4 A sketched chain outline on the *left* and the result of applying GearSketch's tightening algorithm on the *right*

than being able to just press the play button and see it. If learners do not solve the problem on their first try, the play button is then enabled, so they can explore why their solution was incorrect.

These interface features provide the learner with different types of feedback. By aligning gears' teeth and tightening chains, GearSketch ensures that the configurations learners create are valid and represented unambiguously. Creating a gear configuration in this way is a form of externalization (Cox 1999) during which learners are assisted with the help of GearSketch's domain model. The second type of feedback is the animation of the gears when the play button is pressed. A recent meta-analysis (Höffler and Leutner 2007) shows that this type of dynamic representational visualization leads to higher learning outcomes than static visualizations.

Learner model

GearSketch continuously updates a Bayesian network model (Koller and Friedman 2009) of each student's domain knowledge as they answer questions and attempt to solve puzzles. Knowledge tracing (Corbett and Anderson 1995) is used to update the estimates of individual learners' knowledge of each of the rules and of their ability to recognize each rule's applicability in solving puzzles. Figure 5 shows the structure of a small part of the Bayesian network that is used to model learners' knowledge. The learner model contains three nodes: Q, R and P, for each declarative rule in Tables 1 and 2. The Q node indicates the learner's ability to apply the rule in a question context, which follows in a straightforward way from having declarative knowledge of the rule. The R node indicates the learner's ability to recognize the relevance of this rule when solving a puzzle. The P node, the value of which depends on both the Q and the R nodes, indicates the learner's ability to apply the rule in a puzzle context. Each node is assigned a value between 0 and 1, which represents the probability that a learner has the ability to which this node refers. When the values of the parent nodes (those nodes with arrows pointing to the current node) are known, the value of the current node can be calculated automatically. For instance, the probability that a student is able to apply a rule in a puzzle context (P node), can be calculated based on the probability that the learner is able to apply this rule in a question context (Q node) and the probability that the learner is able to recognize the relevance of this rule (R node). The initial values for the parent nodes were based on experimental results that indicated which rules students found it easy or difficult to grasp.

When a learner completes an item, this item is temporarily added to the network with the appropriate connections. The Question item and Puzzle item nodes in Fig. 5 represent the question and puzzle shown in Figs. 1 and 2. Although the same declarative rules are needed to solve these items, Fig. 5 shows that the items are not connected to the same nodes. As discussed earlier, to solve the puzzle item, learners need to know the relevant rule and to recognize its relevance for solving the puzzle. Therefore, the P nodes connected to puzzle items take both of these abilities into account. When the student submits an answer, GearSketch checks whether it is correct and updates the value of the item in the network. Using Bayesian inference (Koller and Friedman 2009), the other nodes in the graph are then updated to reflect this new information about the student's abilities. Then, the new state of the model is saved by copying the new values of the rule nodes and removing the temporary item node, as discussed by Conati et al. (2002). Students do not attempt to solve items just so that the learner model can be updated; they also learn from this activity. Therefore, the final step consists of calculating the probability that the student learned the relevant rules from attempting this item, following the principles discussed by Corbett and Anderson (1995), and then updating the values of the rule nodes.

For example, a learner is asked a simple question about the turning direction of a gear that meshes with another gear that is turning clockwise. Prior to the selection of this question the learner model assigned a probability of 0.6 to this learner knowing the relevant rule. The probability that the learner answers the question correctly is not simply equal to the probability that the learner knows the rule, because the learner could guess the correct answer even when he or she does not know the rule or make a mistake even though he or she does know the rule. These probabilities are referred to as the *guess* and *slip* parameters. Because there are two possible answers (the gear turns clockwise or counterclockwise), it is reasonable to choose 0.5 as the guess parameter. The chance of slipping on this simple question is small, so 0.1 seems a reasonable value for the slip parameter. Using these values, we can now calculate that the predicted probability of the learner answering the question correctly is $0.9 \times 0.6 + 0.5 \times 0.4 = 0.74$. When the learner actually answers the question we have more information, which we can use to update our previous estimate of the probability that the learner knows the relevant rule. For instance, if the learner would answer the question incorrectly, it follows by Bayesian inference that the probability that the learner knew the rule when answering the question was $\frac{0.1 \times 0.6}{0.1 \times 0.6 + 0.5 \times 0.4} = 0.23$. Finally, there is a chance that although the learner did not know this rule prior to answering this question, he or she learned it from attempting this problem and receiving feedback. We call this the *transition* probability. For a transition probability of 0.3 the updated estimated probability that the student knows the rule after (incorrectly) answering this question is equal to $0.23 + 0.3 \times (1 - 0.23) = 0.46$.

GearSketch then uses the updated learner model for the individual student to select his or her next item. Because GearSketch has information about the rules required to solve each item, an item of appropriate difficulty can be selected. The probability of the learner correctly solving a candidate item can be determined by temporarily inserting this item into the learner model and updating its value using Bayesian inference. Students of below average and above average ability can benefit from this type of adaptive problem selection

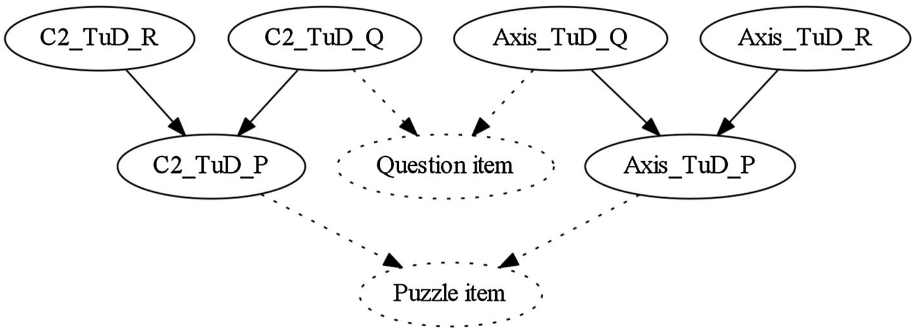


Fig. 5 The structure of a small part of the learner model. *C2* and *Axis* refer to connection types between gears, connected by a chain and through their axes, respectively. The *R* suffix refers to the learner's ability to recognize the relevance of a rule in a puzzle context and the *Q* and *P* suffixes refer to the learner's ability to use the rule in a question and a puzzle context, respectively

compared to static problem selection, because it allows them to learn more efficiently (Mitrovic and Martin 2004).

Questions and puzzles can either be simple or complex. An item is defined as simple if its solution involves application of just one rule and defined as complex when it requires application of multiple rules. To offer students a variety of items, GearSketch alternates between the four item types in this order: simple question, simple puzzle, complex question, complex puzzle. After completing a complex puzzle, the student is given a new simple question. If no suitable item in one of these categories can be found, an item from the next category is selected. The criteria for an item's suitability are straightforward. Simple puzzle items are only suitable when the learner model indicates that there is a high probability that the student can apply the relevant rule in the context of a question. For instance, a student will only be given a puzzle that requires use of the rule about the turning direction of meshing gears, if the probability that they can apply this rule is at least 0.75 according to the learner model. Complex question and puzzle items are only suitable when the learner model indicates that the probability that the student can apply each of the relevant rules in either a question or puzzle context is high. In practice, this means that students must first correctly answer questions about a rule before they are given a puzzle that requires using this rule. Additionally, before students are given complex items, they must show that they can solve simple items for each of the rules needed to solve the complex item. This leads to a gradual, individualized progression of item difficulty.

Abstract and concrete items

Letting students practice applying each of the rules in Tables 1 and 2 in both question and puzzle contexts requires a lot of different items. Instead of exactly specifying each of these items, GearSketch uses abstract item descriptions that are converted into concrete items when they are presented to learners. This means that learners cannot use surface features to recognize items they have attempted multiple times, but must identify the underlying abstract structure of a given gear configuration. An abstract question item contains a question with a set of possible answers and information about the correct answer. An abstract puzzle item contains instructions for the learner regarding the goal of the puzzle and sufficient information for GearSketch to evaluate whether that goal was reached. Both types of abstract items contain an abstract description of the concrete gear configuration that will be shown to the learner. This abstract description is used to generate a new concrete gear configuration each time a student attempts this item. Figure 6 shows an example of an abstract description and two concrete configurations created by GearSketch based on this description. The abstract description specifies information about the gears, their properties and the connections between them. The "unmeshable" and "unstackable" properties are used to create puzzles where certain gears cannot be connected via their teeth or axes. These constraints can be used to guide students in a certain direction or to remove obvious solutions, making puzzles either easier or more difficult.

Experimental evaluation

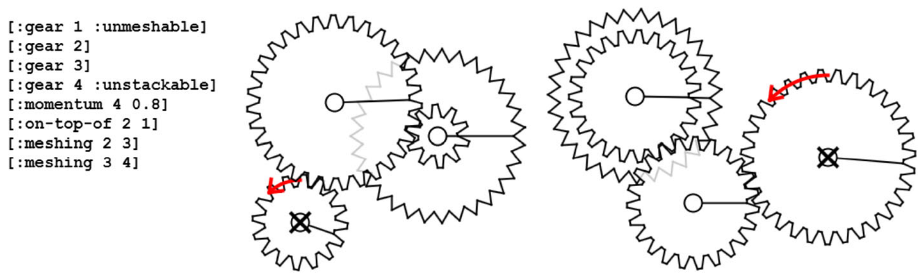
Two experimental studies have evaluated the effectiveness of GearSketch. The first study, described in detail in Leenaars et al. (2012) investigated the value of GearSketch's drawing-based interface, that interprets and animates students' drawings. This study showed that a group of students with access to these features learned significantly more from working with GearSketch than a group of students who worked with a version of GearSketch that lacked these features.

A second study investigated the value of GearSketch's learner model and adaptive features. Two evaluation questions were addressed by this study. First, does tracking individual students' knowledge with a learner model and using this model to select appropriate practice problems help students learn more from working with GearSketch? Second, how accurately does the learner model predict students' posttest results? This study has not previously been reported and will be described here.

Method

Participants in the second study were 44 fifth grade students (26 female), randomly assigned to the experimental (22 students, 12 female) or control condition (22 students, 14 female). The experimental group used a version of GearSketch in which a learner model that was updated after they completed each item was then used to select their next item. The control group worked with a fixed sequence of items, which was created by running a simulation of a student attempting items and updating its learner model to select the next item. When participants started working with GearSketch they first completed 19 short tutorials and then continued answering questions and solving puzzles in GearSketch for a total of 60 min. Both groups were then given the same test, which consisted of a fixed selection of nine questions and six puzzles of varying difficulty. This test was integrated into GearSketch. During the experiment, participants could ask the researcher (male) clarifying questions, but otherwise worked individually. No pretests were used, because students were not yet familiar with terms and concepts like "tooth speed" and "turning speed" that are required to communicate questions and puzzles in the domain. Because the participants were drawn from a homogeneous group and randomly assigned to one of the two conditions, no prior differences between the groups were expected.

We hypothesized that the experimental group would outperform the control group on the test, because the items they practiced with were tailored to the learner model for each individual participant. Therefore, we compared the test scores of the two groups using a



```
[:gear 1 :unmeshable]
[:gear 2]
[:gear 3]
[:gear 4 :unstackable]
[:momentum 4 0.8]
[:on-top-of 2 1]
[:meshing 2 3]
[:meshing 3 4]
```

Fig. 6 An abstract description and two concrete instantiations of a gear configuration

one-tailed *t* test. Additionally, we examined the quality of the learner model's predictions. Correct answers on the test were scored as 1 and incorrect answers as 0. For each test item completed by the experimental group, the predicted probability of each student answering it correctly was also extracted from his or her learner model. Then the correlation between the learner models' predictions and the actual test scores of the experimental group was calculated. Finally, these predictions were compared with the actual test scores using a two-tailed paired samples *t* test, to see whether the learner model's predictions displayed any bias toward overestimation or underestimation of the students' capabilities.

Results

The average test scores for the experimental and control group were 7.50 and 7.36 respectively. There was no significant difference ($t = 0.23$, $d = 0.07$, $p = 0.408$) between the test scores of these groups. The correlation between the learner models' predictions and actual test performance of the experimental group was significant ($r = 0.168$, $p = 0.002$). The predicted test score for the experimental group based on the learner model was 7.49. There was no significant bias towards overestimation or underestimation in the learner models' predictions of students' performance on the test ($t = 0.035$, $p = 0.972$).

Discussion

Contrary to our hypothesis, the experimental group's performance on the test was not significantly better than the control group's performance. Additionally, we found that the learner model's predictions of students' test results were not very accurate, although there was no bias towards overestimation or underestimation. The low correlation between the test scores and the learner model's predicted test scores offers a possible explanation for the finding that tailoring item selection did not improve learning outcomes. Because this learner model's predictions were also the basis for the selection of practice items, it is likely that the selection of practice items was not optimal. Therefore, improving the learner model may be a fruitful approach to making the adaptive item selection of GearSketch more effective in supporting the students' learning process. Such an improvement could potentially be achieved by changing the structure or parameters of the learner model or by letting students complete more practice items to provide the learner model with more data about the students' abilities.

General discussion

This paper discusses the development of GearSketch to answer four development questions. The learning goal for students working with GearSketch is to be able to solve qualitative gear problems, such as connecting gears in such a way that they turn with a given relative speed or in a given direction. To solve these problems, students need to acquire procedural knowledge of the declarative rules found in Tables 1 and 2 and an understanding of relevant spatial constraints (DQ1). To help students acquire this knowledge, GearSketch lets students draw and explore gear and chain configurations by interpreting their pen strokes and animating the resulting model (DQ2). The declarative rules of the domain model are introduced in tutorials and students attempt to solve practice problems to compile these rules into procedural knowledge. The rules needed to solve each of these practice problems are represented explicitly in their abstract descriptions, so that the Bayesian learner model can track each student's progress as he or she works through these questions and puzzles (DQ3). The gear configurations used in the questions and puzzles are described abstractly, in terms of structure instead of absolute position. This lets GearSketch create a new concrete problem each time a student attempts to solve the same abstract problem. Therefore students cannot learn how to solve the problems through memorizing their surface features, but have to pay attention to their structure (DQ4).

The first experimental study, discussed in Leenaars et al. (2012), showed the value of GearSketch's domain model and interface. Students who worked with a version of GearSketch that interpreted and animated their drawings learned significantly more than students who worked with a version of GearSketch that lacked these features (EQ1). A second experimental study did not show that GearSketch's adaptive item selection leads to more efficient learning (EQ2). Because other studies have shown the value of learner models and adaptive item selection in digital learning environments (Desmarais and Baker 2012), this result deserves further investigation. A plausible explanation for this lack of an effect is that the learner model does not adequately represent students' abilities. The learner model's predictions were significantly correlated with students' test scores and were not biased, but the correlation was low (EQ3). This could be due to incorrectness of the initial parameters of the learner model or because the students did not work with GearSketch long enough to provide the learner model with sufficient data to capture their knowledge. Other possible explanations could be that suboptimal questions and puzzles were chosen by the item selection mechanism or that students needed more support from the learning environment when they failed to solve an item in order to really learn from this experience. These candidate explanations can be tested with further analysis and empirical studies.

The current experimental evaluation still has a number of limitations. The evaluation studies did not compare the learning outcomes of working with GearSketch to those of learning about gears using a textbook or doing tasks with physical gears. Therefore it is not yet possible to draw conclusions about possible advantages of using GearSketch compared to the status quo. Furthermore, no pretests were used in the evaluation studies. This makes it difficult to quantify the learning effect of working with GearSketch. However, the mean difference between the posttest scores of the experimental group and the control group in the study reported in Leenaars et al. (2012) does provide an estimate of a lower bound for this learning effect. Because participants were drawn from a homogeneous group and randomly assigned to the conditions and it seems unlikely that the students in the control group experienced a negative learning effect, the learning effect for the experimental group is likely at least as large as the mean posttest score difference between the groups. Future studies will give more insight into the effectiveness of GearSketch's approach.

Although GearSketch is a domain-specific learning environment, the principles used in its design can be applied in other domains as well. Specifically, the approach to sketch recognition and simulation could be applied in other domains consisting of a limited number of rules and elements that can interact in complex ways and having a natural pictorial representation. A good example is the ropes and pulleys domain. More generally, GearSketch's approach to teaching complex problem-solving skills is based on introducing the theory of the domain with hands-on tutorials and offering practice items with which learners can experiment. Thanks to the drawing-based nature of GearSketch, experimentation is as easy as making a sketch and pressing the play button to animate it. Offering students individually tailored items may further support their learning process, but adequately modeling their abilities has so far proven to be challenging.

The increasing availability of pen-based and touchscreen devices in homes and in schools has made possible the use of drawing-based digital learning environments in education. By using these new technologies and focusing on a specific domain, a learning environment like GearSketch can offer an interface and simulations that support learners in finding their own solutions to complex problems in this domain. Additionally, this domain-specific focus lets GearSketch incorporate ideas from cognitive and constraint-based tutors (Desmarais and Baker 2012), which makes it possible to tailor problem selection to individual learners. The same touchscreen technology can be used with more general drawing-based learning environments that can support learners in exploring a large variety of different domains. For example, CogSketch (Forbus et al. 2011) uses open-domain sketch understanding to reason about the drawings that learners make and can give feedback based on this understanding. SimSketch (Bollen and Van Joolingen 2013) allows learners to explore the behavior of models that they construct themselves by drawing. These developments show some of the ways that new technologies allow ideas from drawing-based learning to be combined with ideas from technology-enhanced learning to help students learn how to solve complex problems.

References

Ainsworth, S., Prain, V., & Tytler, R. (2011). Drawing to learn in science. *Science, 333*(6046), 1096–1097. doi:10.1126/science.1204153.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060. doi:10.1037/0033-295X.111.4.1036.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*(2), 167–207. doi:10.1207/s15327809jls0402_2.

Andrade, A. (2009). The clock project: Gears as visual-tangible representations for mathematical concepts. *International Journal of Technology and Design Education, 21*, 93–110. doi:10.1007/s10798-009-9104-x.

Bartolini Bussi, M. G., Boni, M., Ferri, F., & Garuti, R. (1999). Early approach to theoretical thinking: Gears in primary school. *Educational Studies in Mathematics, 39*(1–3), 67–87. doi:10.1023/A:1003707727896.

Bespamyatnikh, S. (2003). Computing homotopic shortest paths in the plane. *Journal of Algorithms, 49*(2), 284–303. doi:10.1016/S0196-6774(03)00090-7.

Bollen, L., & Van Joolingen, W. R. (2013). SimSketch: Multi-agent simulations based on learner-created sketches for early science education. *IEEE Transactions on Learning Technologies,*. doi:10.1109/TLT.2013.9.

Chambers, J. M., Carbonaro, M., & Murray, H. (2008). Developing conceptual understanding of mechanical advantage through the use of Lego robotic technology. *Australasian Journal of Educational Technology, 24*(4), 387–401.

Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction, 12*(4), 371–417. doi:10.1023/A:1021258506583.

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modelling and User-Adapted Interaction, 4*(4), 253–278. doi:10.1007/BF01099821.

Cox, R. (1999). Representation construction, externalised cognition and individual differences. *Learning and Instruction, 9*(4), 343–363. doi:10.1016/S0959-4752(98)00051-6.

Desmarais, M. C., & Baker, R. S. J. d. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction, 22*(1–2), 9–38. doi:10.1007/s11257-011-9106-8.

Dixon, J. A., & Bangert, A. S. (2004). On the spontaneous discovery of a mathematical relation during problem solving. *Cognitive Science, 28*(3), 433–449. doi:10.1207/s15516709cog2803_6.

Forbus, K., Usher, J., Lovett, A., Lockwood, K., & Wetzel, J. (2011). CogSketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science, 3*(4), 648–666. doi:10.1111/j.1756-8765.2011.01149.x.

Höffler, T. N., & Leutner, D. (2007). Instructional animation versus static pictures: A meta-analysis. *Learning and Instruction, 17*(6), 722–738. doi:10.1016/j.learninstruc.2007.09.013.

Hu, W. (2011, January 4). More schools embrace the iPad as a learning tool. *The New York Times*. Retrieved from http://www.nytimes.com/2011/01/05/education/05tablets.html.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques* (1st ed.). Cambridge: The MIT Press.

Lee, M. (2010). Interactive whiteboards and schooling: The context. *Technology, Pedagogy and Education, 19*(2), 133–141. doi:10.1080/1475939X.2010.491215.

Leenaars, F. A. J., Van Joolingen, W. R., & Bollen, L. (2013). Using self-made drawings to support modelling in science education. *British Journal of Educational Technology, 44*(1), 82–94. doi:10.1111/j.1467-8535.2011.01272.x.

Leenaars, F., Van Joolingen, W., Gijlers, H., & Bollen, L. (2012). Drawing-based simulation for primary school science education: An Experimental study of the GearSketch learning environment. In *2012 IEEE Fourth International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL)* (pp. 1–8). doi:10.1109/DIGITEL.2012.9.

Mitrovic, A. (2003). An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education, 13*(2–4), 173–197.

Mitrovic, A., & Martin, B. (2004). Evaluating adaptive problem selection. In P. M. E. D. Bra & W. Nejdl (Eds.), *Adaptive hypermedia and adaptive web-based systems* (pp. 185–194). Berlin: Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-27780-4_22.

Mitrovic, A., Mayo, M., Suraweera, P., & Martin, B. (2001). Constraint-based tutors: A success story. In L. Monostori, J. Váncza, & M. Ali (Eds.), *Engineering of intelligent systems* (Vol. 2070, pp. 931–940). Berlin: Springer.

Van Essen, G., & Hamaker, C. (1990). Using self-generated drawings to solve arithmetic word problems. *The Journal of Educational Research, 83*(6), 301–312.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education, 15*(3), 147–204.

Frank A. J. Leenaars studied Computer Science (B.Sc.) and Psychology (B.Sc., M.Sc.) and is currently a Ph.D. student at the department of Instructional Technology at the University of Twente. He works on the development and evaluation of innovative digital learning environments for primary school science education using an approach based on drawing and modeling.

Wouter R. van Joolingen is professor of dynamical modeling in educational settings. His work concerns the use of simulations and modeling in science education. His main interest is making computer modeling accessible to young children with the help of intuitive means of model specification, such as drawings and qualitative models.

Hannie Gijlers studied Educational Science at the University of Groningen. She received her Ph.D. from the University of Twente where she worked on computer supported collaborative inquiry learning in science. She continues her research in the field of collaborative inquiry learning. The focus of her current research project is on the interaction between cognitive and communicative processes in collaborative inquiry learning environments.

Lars Bollen received his Ph.D. in 2009 at the University Duisburg-Essen on “Activity Structuring and Activity Monitoring in Heterogeneous Learning Scenarios with Mobile Devices”. Currently, he works at the University of Twente on the EU-project SCY and on the project “Thinking with Inaccurate Drawings”. His research interests include model-based learning & sketching, pen-based devices in learning scenarios, and (inter)action analysis.