Scaffolding Learning by Modelling: The Effects of Partially Worked-Out Models

Yvonne G. Mulder, Lars Bollen, Ton de Jong, and Ard W. Lazonder

Department of Instructional Technology, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Received 31 December 2013; Accepted 13 May 2015

Abstract: Creating executable computer models is a potentially powerful approach to science learning. Learning by modelling is also challenging because students can easily get overwhelmed by the inherent complexities of the task. This study investigated whether offering partially worked-out models can facilitate students’ modelling practices and promote learning. Partially worked-out models were expected to aid model construction by revealing the overall structure of the model, and thus enabling student to create better models and learn from the experience. This assumption was tested in high school biology classes where students modelled the human glucose-insulin regulatory system. Students either received support in the form of a partial model that outlined the basic structure of the glucose-insulin system (PM condition; \( n = 26 \)), an extended partial model that also contained a set of variables students could use to complete the model (PM+ condition; \( n = 21 \)), or no support (control condition; \( n = 23 \)). Results showed a significant knowledge increase from pretest to posttest in all conditions. Consistent with expectations, knowledge gains were higher in the two partial model conditions than in the control condition. Students in both partial model conditions also ran their model more often to check its accuracy, and eventually built better models than students from the control condition. Comparison between the PM and PM+ conditions showed that more extensive support further increased knowledge acquisition, model quality, and model testing activities. Based on these findings, it was concluded that partial solutions can support learning by modelling, and that offering both a structure of a model and a list of variables yields the best results. © 2015 Wiley Periodicals, Inc. J Res Sci Teach 53: 502–523, 2016

Keywords: learning by modelling; completion problems; worked-out examples

Dutch summary (Nederlandse samenvatting): Modelleren is een veelbelovende manier voor het leren van natuurwetenschappelijke principes en verschijnselen. Leren door te modelleren is echter niet eenvoudig: het maken van een werkend computermodel blijkt voor veel leerlingen een struikelblok te zijn. Het aanbieden van een gedeeltelijk uitgewerkt model zou deze problemen kunnen doen verminderen. In een gedeeltelijk uitgewerkt model wordt de algemene modelstructuur gegeven, wat de leerlingen kan helpen om een beter model te maken en de inhoud ervan beter te begrijpen. Deze hypotheses zijn onderzocht tijdens een biologiepracticum waarin leerlingen uit de bovenbouw van het VWO een computermodel van het glucose-insuline regulatieproces moesten maken. Een deel van de leerlingen had hierbij de beschikking over een gedeeltelijk uitgewerkt model (PM conditie; \( n = 26 \)). De tweede groep kreeg hetzelfde uitgewerkte model plus een lijst met variabelen die zij aan het model konden toevoegen (PM+ conditie; \( n = 21 \)), terwijl de
leerlingen uit de controle conditie \((n = 23)\) geen extra ondersteuning kregen. Uit de resultaten op de natoets bleek dat de leerlingen uit alle drie de condities iets hadden geleerd over glucose-insuline regulatie. De kennisstoename was echter significant hoger in de beide condities met het gedeeltelijk uitgewerkte model. De leerlingen uit deze condities hebben hun modellen bovendien vaker laten doorrekenen en eindigden het practicum met een significant beter model dan de leerlingen uit de controle conditie. Uit de vergelijking tussen de PM en PM+ conditie bleek verder dat de lijst met variabelen tot een grotere verbetering van het leerproces en de leersoutkomsten leidt. Gedeeltelijk uitgewerkte modellen zijn dus effectief als ondersteuning voor het leren door te modelleren, vooral wanneer naast een algemene modelstructuur ook een lijst met variabelen wordt gegeven.

Trefwoorden: leren door te modelleren; completeerproblemen; uitgewerkte voorbeelden

Learning formal disciplines such as science often involves the development of cognitive models of a phenomenon, topic, or process. Students’ cognitive model construction can be greatly advanced by using external representations, which can take the form of executable computer models when the learning content is of a dynamic nature. Well-known examples include animations that help visualize the fundamental biological processes of photosynthesis or cell division, and simulations that enable students to explore predator-prey ecosystems. These applications are indicative of what de Jong and van Joolingen (2007) termed learning from models: students receive a predefined model they can inspect or use to explore the underlying properties of a scientific phenomenon. Another, less well-known alternative is learning by modelling, which refers to an instructional regime where students are asked to construct an executable model of the scientific phenomenon themselves.

The potential of learning by modelling is increasingly recognized by science teachers and policy makers worldwide (CCSSO, 2013; Harlow, Bianchini, Swanson, & Dwyer, 2013; Henze, van Driel, & Verloop, 2007; van Dijk, Hajer, Scharten, & de Vos, 2013). This is perhaps best illustrated by the fact that the construction and use of models is featured prominently in the Next Generation Science Standards for K-12 science education in the United States (National Research Council, 2012). Educational researchers acknowledge that modelling can be a valuable yet challenging pedagogical approach to science learning (e.g., Halloun, 2006; Louca & Zacharia, 2012; Schwarz & White, 2005). To help students and their teachers overcome the inherent complexities of learning by modelling, several attempts have been made to make modelling practices accessible and meaningful for students in K-12 classrooms. A notable example is the Modeling Designs for Learning Science project, which proposed guidelines for the design of curriculum materials that enable upper elementary and middle school students to engage in progressively more sophisticated modelling practices (Schwartz et al., 2009).

The literature indicates that learning by modelling distinguishes itself from other approaches to science learning in that it involves students in a range of deep cognitive processes such as analysing, relational reasoning, synthesizing, and testing and debugging (Jackson, Stratford, Krajcik, & Soloway, 1994; Louca & Zacharia, 2015; Schwartz et al., 2009), which eventually leads to a profound understanding of the topic at hand (Hashem & Mioduser, 2011; Hmelo-Silver, Liu, Gray, & Jordan, 2015). Louca and Zacharia (2012) organized these cognitive processes according to the two broad stages students go through when modelling a scientific phenomenon. During the model formulation stage, students develop a model of a phenomenon or system by identifying key elements and creating links that indicate how these elements are related. In the model deployment stage, students test their model in new situations and adapt it when necessary on the basis of acquired results. Students’ modelling practices thus mimic authentic scientific inquiry: students can use models to express their understanding by defining relevant variables and
their relations, and run the model to verify the accuracy of their propositions (Jackson et al., 1994; White, Shimoda, & Frederiksen, 1999).

Engaging in these modelling practices can be beneficial for science learning. It enables students to develop a better understanding of the behavior of systems in general (Hmelo-Silver et al., 2015; Hogan & Thomas, 2001; Stratford, Krajcik, & Soloway, 1998; van Borkulo, 2009), promotes the development of specific (scientific) reasoning skills (Milrad, Spector, & Davidsen, 2003), and leads to more profound domain-specific knowledge. To illustrate, building models of complex scientific phenomena can offer students the opportunity to learn about explanatory frameworks that make sense of phenomena at multiple levels (Wilensky & Resnick, 1999). These phenomena often depend on local interactions of components of the systems, and the function of the entire system may appear quite different from the behavior of individual elements (Hmelo, Holton, & Kolodner, 2000). In this context, Wilensky and Resnick (1999) provide the example of a traffic jam that moves backward even though the cars in the traffic jam are going forward. During learning by modelling, students can explore how changes at the level of the local interactions of components in a system lead to different behaviors and patterns at another level, namely the function of the entire system, thus implicitly learning about complex systems and emergent behavior. Studies comparing learning by modelling with other, more expository forms of instruction indicate that learning by modelling positively fosters this more advanced reasoning about structures (Hansen, Barnett, MaKinster, & Keating, 2004; van Borkulo, van Joolingen, Savelsbergh, & de Jong, 2012).

However, these benefits only apply if students receive adequate guidance. Research in related areas such as learning from simulated models shows that students need support to overcome the problems they encounter when applying the cognitive processes identified above (de Jong, 2006; Minner, Levy, & Century, 2010). The literature on modelling practices supports the notion that students experience difficulties when using and creating models (de Jong & van Joolingen, 1998; Hogan & Thomas, 2001). In the model formulation phase, students often fail to relate their knowledge of phenomena to the models, even though they are capable of building syntactically correct models (Sins, Savelsbergh, & van Joolingen, 2005). When building their model, students often ignore the behavior of the system as a whole (Hogan & Thomas, 2001; Sins et al., 2005). The weaker students in particular tend to focus on one quantity at a time, and are thus hindered by the details of the model right from the start. The more successful students take the overall model structure into account and include the most salient parts of the model first. Studies comparing novice and expert modelling behavior indicate that experts spend a long time thinking through the entire model in the model formulation stage (Wu, Wu, Zhang, & Hsu, 2013; Zhang, Liu, & Krajcik, 2006). They thoroughly consider which elements to include before creating the relations between these elements. Similar to experts, students are quite capable of identifying key elements —although they occasionally include irrelevant elements in their models—but have more difficulties in identifying relationships than the experts (Mulder, Lazonder, & de Jong, 2010; Wu, 2009). Students therefore need a relatively long time to create a model structure (Mulder, Lazonder, & de Jong, 2011; Mulder, Lazonder, de Jong, Anjewierden, & Bollen, 2012), leaving little time for the deployment of the model in new situations.

This lack of time could be one reason why many students exhibit poor performance in the model deployment phase, where they generally fail to engage in dynamic iterations between examining output and revising models (Hogan & Thomas, 2001), and lack persistence in debugging models to fine-tune its behavior (Stratford et al., 1998). An alternative reason might be that students tend to overestimate the quality of their models (Mulder et al., 2011, 2012). These studies show that students often consider their model to be complete when it is in fact still inaccurate. This tendency poses a problem, considering that the benefits of learning by modelling
only hold when students create accurate models (Alessi, 2000). Creating mediocre models might engage students in thinking about the model structure, but fails to provide them the opportunity to inspect the dynamic behavior of the modelled system. Together these findings substantiate that students need support to address and overcome these difficulties.

**Supporting Learning by Modelling**

Support for student-directed forms of learning often takes the form of scaffolding. The term “scaffolding” was introduced by Wood, Burner, and Ross (1976) to indicate the process by which a more knowledgeable person helps a learner succeed in tasks that would otherwise be beyond his/her reach. Researchers in the learning sciences soon embraced the conception and gradually extended its application to situations where support is offered by a more capable peer, tools within the learning environment, or both (Puntambekar & Hübscher, 2005; Tabak, 2004). Another extension concerned the purpose of scaffolding: it should not only assist learners in accomplishing tasks, but also enable them to learn from the experience (Reiser, 2004).

The focus in this article is on software scaffolding. Environments for inquiry-based and model-based science learning can be equipped with designated tools and facilities that assist students in completing a task (for an overview, see Quintana et al., 2004). For example, the learning environment could generate prompts to remind students of important steps in the learning process (Wichmann & Leutner, 2009), give hints to help them perform these steps (Slotta & Linn, 2009), explain important domain concepts (Lazonder, Hagemans, & de Jong, 2010; Wecker et al., 2013), or reduce the complexity of the task by restricting the number of options students need to consider (Mulder et al., 2011). These software scaffolds support students by structuring the learning task and guiding them through the core steps. Such scaffolding clearly serves to assist students in accomplishing the task, but does not appear to directly address students’ learning from this experience. According to Reiser (2004); software tools can shape students’ understanding by problematizing subject matter. Examples of this mechanism include pointing out flaws or inconsistencies in the students’ work, challenging students to reconsider their work from a different perspective, or encouraging them to attend to aspects of the task they might otherwise overlook. Effective scaffolds should thus structure the task to make it more tractable for students while at the same time problematize the task content to draw students’ attention to issues that will be productive for learning.

Research on learning by modelling has mainly examined scaffolding techniques that aim to enhance students’ performance during model formulation activities. For instance, Löhner, van Joolingen, and Savelsbergh (2003) showed that students benefit from a graphical modelling language that makes use of the traditional “stock and flow” language (Forrester, 1961) compared to a text-based language. Students’ model formulation also benefits from constraining scaffolding approaches that organize the task according to a simple-to-complex sequence in order to prevent them from getting overwhelmed by the complexity of the task (e.g., Mulder et al., 2011, 2012; Zhang et al., 2014). The success of such incremental approaches might derive from the fact that students can focus on the model structure outline first before considering the models’ behavior.

However, the Mulder et al. (2011, 2012) studies also showed that even students in the best-performing scaffolded groups still produced mediocre models. And even if these students had created high-quality models, their performance success would not guarantee that they actually learned something from their modelling experiences. To illustrate, tutoring and meta-tutoring systems that provide students with hints and feedback can enhance students’ performance during the modelling task, but not their performance on a knowledge test administered after the task (e.g., Roscoe, Segedy, Sulcer, Jeong, & Biswas, 2013; Zhang et al., 2014). In other words, scaffolds that help students create better models are not necessarily the ones that help them learn about the
system they are modelling. It appears that there can be a trade-off between supporting modelling
performance in the short term and learning in the long term. Therefore research should look for
effective scaffolding approaches that enable students both to create accurate models and to
develop a profound understanding of the underlying domain content.

Some useful guidance on this issue can be gleaned from research on worked-out examples.
Scaffolds derived from the example-based learning approach can enhance students’ performance
success as well as increase their levels of conceptual knowledge (for a recent overview, see Renkl,
2011). Such worked-out examples essentially include a problem statement, a step-by-step account
of the procedure for solving the problem, and the final solution. The direct availability of useful
worked-out examples has long since been found to reduce the number of errors and improve both
near and far transfer (Sweller & Cooper, 1985). We think that the essence of worked-out examples
can be applied, in an adapted way, to learning by modelling. Learning by modelling revolves
around the students’ creation of an integrated model of a problem domain. This means that
students cannot be given a series of worked-out models; doing so would readily give them the
solution to the task at hand and make the model formulation phase redundant. What can be done,
however, is to give students a partially worked-out model, similar to the partially worked-out
problems that are used in example-based learning (van Merrienboer, 1990).

Having students complete a partial solution is recognized as a powerful scaffolding technique
(Sweller, van Merrienboer, & Paas, 1998). These so-called completion problems combine the
strengths of worked-out examples and conventional problem solving: they provide (part of) a
worked example while at the same time requiring students to solve the problem. Early research on
the instructional benefits of completion problems was conducted in the fields of statistical problem
solving and computer programming. Paas (1992) showed that completion problems and fully
worked-out examples have a comparable positive effect on learning to solve statistical problems:
students who either completed practice problems or studied worked-out examples outperformed
students who solved the practice problems from scratch. Similar results were found by van
Merrienboer (1990) in the field of programming instruction. He found that completion
assignments led to higher learning outcomes than generating a program from scratch. In a follow-
up study, van Merrienboer and de Croock (1992) compared the learning activities related to
completion and generation assignments. They found that completion problems reduce help-
seeking efforts, evoke testing and debugging activities, and they replicated the finding that
completion problems generally lead to better performance. More recently Baars, Visser, van Gog,
Bruin, and Paas (2013) showed that completion problems can also effectively reduce students’
overestimation of their performance. When learning from text, the students in this study who
completed practice problems performed comparably to students who learned from fully worked-
out examples, but had lower and more accurate performance judgements.

Two studies replicated the completion problem findings for concept mapping problems, a
learning task that bears a strong resemblance to the creation of executable models. Chang, Sung,
and Chen (2001) implemented completion problems as a partial concept map in which some nodes
and links were reserved as blanks. These partial solutions were assumed to provide novice students
with a referent knowledge structure for the task domain, which was the biology topic of
reproduction. Results confirmed that completing a partial concept map leads to more accurate,
complete concept maps and higher learning outcomes than constructing a concept map from
scratch. In a follow-up study, Chang, Sung, and Chen (2002) implemented a series of partial
solutions that faded over time in an instructional unit on text comprehension. The fading
concerned the extent to which the partial solutions contained the referent structure, and whether
all relevant elements were given in a concept list. Using a longitudinal design, Chang et al.
compared this partial solution instruction with generating a concept map from scratch, correcting
an existing erroneous concept map, and a no concept mapping control condition. They found favorable effects of the partial solution instruction on summarizing skills. Although elegant in its longitudinal approach, this study gave information neither on the quality of each completed concept map, nor on learning outcomes. Such information would be valuable for tailoring partial solution instruction to novice learners’ needs.

Examining the Effectiveness of Partial Models

The present study investigated the effects of scaffolding domain novices on a modelling task with partial models. These partial models served as a framework for model construction by outlining the structure of the model students had to build. Consistent with Reiser’s (2004) first scaffolding mechanism for software tools, a basic model outline reduces complexity and choice by providing additional structure for the task, which enables students to perform their modelling assignment successfully. Partial models also act to problematize subject matter, a mechanism that, according to Reiser, enhances students’ learning of the system or phenomenon they are modelling. Partial models can problematize subject matter in three ways. First, they focus students’ attention on parts of the model that still need to be specified so that students have to determine which elements are still missing and how they should be linked to the elements already in place. Partial models can also engage students in the modelling task by arousing their curiosity. For example, students may wonder why two variables in the partial solution are not related, or why a particular variable is specified as a constant that does not change over time. Third, partial models can create a cognitive conflict when students’ understanding of the domain contradicts the content of the partial model they receive. In each of these ways, the partial model can stimulate students to scrutinize the model content further, which, in turn, provides valuable opportunities for learning.

The goal of the present study was to determine whether partial models promote learning by modelling in ways predicted by the scaffolding mechanisms of structuring and problematizing. To be more precise, the study compared the performance success, learning outcomes, and model testing activities of students from three experimental conditions. Students in the control condition had to build a model of glucose-insulin regulation from scratch. Students in the partial model (PM) condition worked on the same task with the aid of a given partial model that provided the overall structure of the model. Students in the extended partial model (PM+) condition received the same partial model plus an additional list of variables that should be included in the model.

As the mechanisms of structuring and problematizing operate in different and sometimes opposing ways, two possible scenarios were envisioned. The first assumed that the structuring mechanism would prevail. In that case, partial models would help students to first consider the overall structure of the model and its behavior without being distracted by all the details of the model right from the start. Identifying the relevant variables is one of these details that students might spend a lot of time on, even though it is probably not related to students’ learning of the domain (Mulder, Lazonder, & de Jong, 2014). Thus, providing students with a partial model, and the extended partial model in particular, would presumably leave more time for model deployment. In addition, the research on completion problems suggests that partial models would promote students’ model testing behavior in both stages of the modelling process, and as a result, enhance the quality of their models. In particular, students with an extended partial model would thus enter the model deployment stage sooner and with a better model. This would leave them with more time to find out how a (reasonably) accurate model behaves in different circumstances, which would ultimately lead to a better understanding of the domain.

The second scenario, in contrast, capitalized on the problematizing mechanism. It predicted that partial models would motivate students to thoughtfully complete the basic model structure and fully understand the content of their final model. Students would thus spend a considerable
amount of time in the model formulation stage trying to create a perfect model, which presumably involves extensive testing and debugging. As a result, partial models would leave little time for model deployment, but would allow students to enter this stage with an accurate model and a solid understanding of its elements and behavior. In this scenario too, the presumed effects would probably be most apparent in the extended partial model condition.

Method

Participants

Seventy fourth-graders from five Dutch high school biology classes participated in the experiment. The sample consisted of 41 females and 29 males, with a mean age of 15.46 years ($SD = 3.30$). All participants had taken the required physics, chemistry, and biology courses since starting high school; their science content knowledge was generally comparable to that of junior high school students in the United States.

The experiment was conducted as part of a regular biology unit on hormones, which was taught by a different teacher in each of the five classes, but using exactly the same instructional materials. The teachers confirmed that the specific subject matter addressed in the experiment (i.e., glucose-insulin regulation) had not yet been taught in these classes. The teachers further verified that their students had no prior experience with modelling, as the topic is first introduced in the advanced science courses in fifth and sixth-grade, and also was not addressed in any extracurricular activity.

Still, these preliminary checks cannot rule out the possibility that some students could be more knowledgeable about glucose-insulin regulation than others, for example, because they have friends or relatives with diabetes. We therefore administered a prior knowledge test, which is described in the next section, and arranged the scores within each class from highest to lowest. Class-rank scores were used to ensure that all three experimental conditions contained students with comparatively high, average, and low prior domain knowledge. Students who took the prior knowledge test but either were absent during the remaining part of the experiment or had incomplete data due to malfunctions in the action logging software, were excluded from the sample. This led to 23 students in the control condition, 26 students in the PM condition, and 21 students in the PM+ condition.

Materials

Instructional Text. Students in all three conditions received a six-page instructional text (1,997 words), which they had to read prior to the modelling assignment and could consult during the assignment. The text described the “supply and demand” mechanisms that ensure that the cells in the human body receive blood that contains the right amount of sugar. The text was organized into four sections that addressed (1) how glucose is produced and how much should be in our blood; (2) how the organs in our body balance the glucose level by secreting insulin; (3) how the brain controls this regulation process; and (4) how exactly this regulation process proceeds over time. Students could infer all relevant information on the structure of the model from this text.

Modelling Assignment. All students were assigned to build a model of the glucose-insulin system described in the instructional text. This topic is highly complex and multi-layered in that it involves a system that changes over time, which makes it appropriate for System Dynamics modelling (Forrester, 1961). The assignment was divided into two stages. During the model formulation stage, students had to construct (or in case of the partial model conditions, complete) their model. For this stage, students were given a scenario of a homeostatic state to consider, where
the blood glucose level reaches an equilibrium. During the model deployment stage, two new scenarios were given that required students to think about increasingly complex aspects of the glucose-insulin regulation process for which they could examine their models’ behavior. Both scenarios were intended to trigger testing and debugging activities. The first scenario dealt with eating a pizza, which creates a spike of glucose in the bloodstream; the second concerned Type 1 diabetes, where the body cannot control the blood glucose level.

Modelling Environment. All participants worked with the SCYDynamics modelling environment (de Jong et al., 2010) that housed a model editor, a bar chart, and a graph tool. The model editor tool enabled participants to represent their knowledge of the glucose-insulin system in an executable computer model. As can be seen from Figure 1, the editor displays a model in the System Dynamics formalism (Forrester, 1961) that has a graphical structure consisting of variables and relations. Variables are the constituent elements of a model and can be of three different types: variables that do not change over time (i.e., constants), variables that specify the integration of other variables (i.e., auxiliaries), and variables that accumulate over time (i.e., stocks). Relations define how two or more variables interact. Each relation is visualized by an arrow connector to indicate the causal link between model elements and can be further specified by selecting a pre-defined, qualitative relation from a drop-down menu.

To facilitate students’ model building activities, the constraining principle of model order progression was used to divide the model formulation stage into two successive phases (cf. Mulder et al., 2011; White & Frederiksen, 1990). During the model sketching phase, students had to create the model structure by indicating the elements and the relations between elements (but not specify these relations). When the model structure had been created, students entered the qualitative modelling phase, where they had to specify the relations between the elements in the model in a qualitative manner (e.g., linear increase, curvilinear decrease). The qualitative modelling features were based on an expert reference model of the domain, which remained hidden from the students, that was used to create a mathematically sound and runnable model. The students’ qualitative specifications were internally replaced with correct mathematical formulas and variable values to create meaningful output in the form of graph diagrams. This feature enabled students to execute their model without having to specify relations quantitatively by entering mathematical equations. As quantitative modelling is rather demanding and beyond the scope of the students’ biology curriculum, it was not included in the present study.

Figure 1. The modelling environment with the partially worked-out model. Students in the PM condition started with the partial model that displayed the major components (left pane); students in PM+ condition received the same partial model plus a list of variables (right pane). Students in the control condition started with a blank screen. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com]
The modelling tool enabled participants to test their understanding by running the model and analyzing its output in a bar chart tool and a graph tool. The bar chart provided feedback on the model structure by displaying stacked columns that showed the number of correct, incorrect, and unnamed elements as well as the number of correct and incorrect relations, which was mainly relevant in the model sketching phase. In the qualitative modelling phase, where the model becomes executable, students could use the graph tool to view its behavior over time. The output of these two tools closely resembled the 2-D bar charts and the line charts produced by commercial spreadsheet packages such as Microsoft Excel; a trial version of the modelling environment that houses both tools is available at http://modeldrawing.eu/our-software/scydynamics/.

Partial Models. In the PM condition the model editor contained a partial model that comprised the main components of the model students had to develop (see Figure 1). The variables “blood glucose level” and “insulin level” were included because they represent the core concepts of the glucose-insulin regulatory system; the inflow and outflow arrows were added to indicate that the levels of both substances can increase or decrease over time. The partial model thus provided students with a rudimentary model structure that they could complete by specifying when insulin is produced and how it helps maintain normal levels of glucose in the blood. The design considerations underlying the development of this partial model were derived from empirical and practical evidence. Drawing on previous work on model progression (Mulder et al., 2011), it was deemed more effective to provide students with a global outline of the entire model instead of with a fully specified part of the larger model. This top-down approach also mirrors what experts and successful learners do in the model formulation stage: they first determine the overall structure of the model before going into the model details (Wu et al., 2013; Zhang et al., 2006). Another, more practical reason for providing these model components was that participants in our previous studies often struggled with the concept of inflow and outflow, and found it difficult to determine to which variables these relation arrows should be attached.

Students in the PM+ condition received the same partial model plus a set of five additional variables they could add to the model. All five variables were needed to develop a fully correct model—offering incorrect or irrelevant variables would be misleading and hence inconsistent with the tenets of giving a partial solution. The decision to provide the full set of relevant variables was prompted by the fact that novice modellers spend a considerable amount of time on identifying these variables themselves. Although they eventually succeed in this endeavor, they learn significantly less from identifying variables than from determining how these variables are related (Mulder et al., 2010).

The modelling assignment instructed students from both partial model conditions to complete their partial model. Students in the PM+ condition received no additional information on the correctness, completeness, and relevance of the given variables and had to decide for themselves whether these elements should be included in the model. Students in the control condition did not receive a partial model. They thus started with a blank screen in the model editor and were instructed to build their model from scratch.

Knowledge Test. A nine-item paper-and-pencil test addressed the key domain concepts, the local interactions of the components in the system, and the function of the entire system. Students’ knowledge of the domain concepts (i.e., glucose and insulin) was assessed by two open-ended questions about the function of these substances in the human body. Students’ local model reasoning is related to students’ actions during the model formulation stage and reflects students’ knowledge of the model structure. It was assessed by four items, each addressing a relation that was described in one of the instructional text’s sections. Students could answer these questions by
drawing the shape of the relation in a graph. Students’ system model reasoning is related to students’ actions during the model deployment stage and reflects students’ knowledge of the model behavior. It was assessed by three items that indicated their knowledge of glucose-insulin regulation in the three scenarios (i.e., in homeostasis, when eating high-calorie food, and with Type 1 diabetes). Examples of the three types of items are given in Figure 2. The knowledge test was administered on two occasions: as a pretest before students engaged in their modelling activities, and as a posttest to indicate knowledge gains that resulted from these activities.

A rubric was developed to score participants’ answers as true or false. One point was allocated to each correct response, which led to a maximum score of nine points. Two raters used this rubric to score a set of 25 randomly selected pretests; inter-rater reliability was 0.92 (Cohen’s $\kappa$).

**Procedure**

Students in each class attended two experimental sessions: a 45 minute introduction and a 120 minute practical. Both sessions took place during the students’ regular biology lessons and were guided by the first author (hereafter, the experimenter). Students started the introductory session by completing the knowledge test, which took them approximately 10 minutes. Next, the experimenter gave a short introduction to System Dynamics models, and handed out a tutorial manual students had to work through individually on their computers. This tutorial served to familiarize students with the System Dynamics modelling language and the operation of the SCYDynamics modelling environment. Students in all three conditions received the same tutorial and worked with the same version of the modelling environment. The experimenter was available to give help or answer questions.

The second session took place several days later, depending on the students’ regular schedule. Students were seated at a computer on which the version of the modelling environment for their experimental condition was installed. They received a paper copy of the instructional text and were asked to study this material carefully. When a student indicated that she/he has finished

**Figure 2.** Examples of the three types of items on the knowledge test.
reading, which was generally after about 10 minutes, the experimenter would give him/her the modelling assignment. The student then started SCYDynamics and worked on the assignment for a maximum of 90 minutes. Students could consult the instructional text during the assignment, but were not allowed to ask the experimenter for help. After 45 minutes of hands-on activity, the experimenter reminded the students that the assignment consisted of a model formulation and a model deployment stage. Students continued working on their model for another 45 minutes, but could stop ahead of time if they had completed the assignment. Immediately after completing work on the assignment, students once again took the knowledge test.

**Measurements**

Data were collected to analyze differences between the three conditions with regard to learning outcomes, model testing activities, and performance success. Learning outcomes concerned students’ understanding of glucose-insulin regulation, and were indicated by the increase in scores from the knowledge pretest to the knowledge posttest. Students’ answers to the nine items on the knowledge test were scored as true or false, leading to an overall maximum score of nine points per test. Maximum scores for domain concept knowledge, local model reasoning, and system model reasoning were two, four, and three points, respectively. Model testing activities were assessed from the log files that were generated by the SCYDynamics modelling environment during the second session. These log files contained information on the number of times students ran their model and plotted its output in a bar chart or a graph. The bar chart run score indicated the frequency with which students clicked the “Run” button to consult the bar chart, and thus represented the number of times students requested feedback on their models’ structure. The graph run score indicated how often students clicked the “Run” button to plot a graph, and thus represented the number of times students requested feedback on the model’s behavior.

The log files also kept a digital record of how the students’ models developed over time. The quality of these models was used as measure of performance success. A software agent was designed to calculate a *model score* based on the similarity between the students’ best model and the reference model depicted in Figure 3. The agent recognized specified model variables and the relations between these variables. The agent was sensitive to alternative terms for variables, variant forms of spelling, and used the Levenshtein distance (Levenshtein, 1966) to correct for orthographic mistakes. The agent computed the model score using an adapted version of the rubric developed by Manlove, Lazonder, and de Jong (2009), which has Cohen’s κ reliability estimates in excess of 0.90. Students received one point for each correctly named variable and an additional point if that variable was of the correct type. For relations, one point was awarded for each correct link between two variables. Up to two additional points could be earned if the direction and type of the relation were correct. To establish inter-rater agreement with the software agent, the first author coded all concepts that students included in their models for a subset of 49 participants, yielding a Cohen’s κ of 0.94. To compensate for the initial score for the correct elements already in place in the partial solutions, the model score was converted to a percentage score. For the control condition, a 100% correct score corresponded to the 47 points of the reference model: all nine correctly named variables of the right type; all four correct relations linked to the flows coming into or out of the stocks, which thus had no relation type; and all of the remaining seven correct relations that were in the right direction and of the correct type. In the PM condition, where two elements worth two points each were given, students could only score 43 points on their own, corresponding with a 100% score in this condition. Finally, since all elements were given in the PM+ condition, the number of additional points for a 100% score was 33.
Results

Table 1 summarizes the results for the main variables in this study. Scores on the knowledge pretest showed that students answered fewer than one-third of the items correctly, which indicates that they had little prior knowledge of the glucose-insulin system. Multivariate Analysis of Variance (MANOVA), using Pillai’s trace, was used to check the \( a \ priori \) similarity of the conditions on this measure. Results confirmed that there were no significant differences in prior knowledge between conditions, \( V = 0.09, F(6,132) = 1.07, p = 0.385 \).

A mixed-design MANOVA was performed to analyze the knowledge gains from pretest to posttest in the three conditions. This analysis revealed significant multivariate effects for the within-subject factor Time, \( V = 0.61, F(3,65) = 33.43, p < 0.001 \), the between-subject factor Condition, \( V = 0.20, F(6,132) = 2.40, p = 0.031 \), and for the Condition × Time interaction, \( V = 0.19, F(6,132) = 2.28, p = 0.040 \). Subsequent univariate ANOVAs were used to break down this interaction effect. Results showed no significant effect for knowledge of domain concepts, \( F(2,67) = 0.71, p = 0.495 \), and for local reasoning, \( F(2,67) = 1.32, p = .273 \). However, there was a significant effect for system reasoning, \( F(2,67) = 3.50, p = 0.036 \). Helmert planned contrasts further revealed that the partial model conditions combined improved system reasoning more than the control condition, \( t(67) = -3.33, p = 0.001, r = 0.38 \), and that these knowledge gains were higher in the PM+ condition than in the PM condition, \( t(67) = 1.98, p = 0.052, r = 0.24 \).

Having established that partial solutions promote better learning outcomes, additional analyses sought to reveal their effects on performance success. The model score indicates the degree of similarity between the students’ models and the reference model. Raw scores were

---

Figure 3. Reference model of the human glucose-insulin regulatory system used to evaluate the quality of the student models.
converted into percentages to compensate for the number of correct elements that were provided in the partial solution conditions. ANOVA indicated a significant between-group effect on this score, $F(2,67) = 82.19, p < 0.001$. Helmert planned contrasts showed that the model score in both partial model conditions was higher than in the control condition $t(67) = -5.47, p < 0.001, r = 0.56$, and that students in the PM+ condition created comparatively better models than those in the PM condition, $t(67) = -11.91, p < 0.001, r = 0.82$.

Students could engage in model testing activities such as running the model and checking its accuracy by inspecting the output in a bar chart or a graph. MANOVA on these activities produced a significant effect for condition, $V = 0.67, F(4,134) = 16.88, p < 0.001$. Subsequent univariate ANOVAs showed significant between-group differences for both types of model runs (bar chart runs: $F(2,67) = 8.74, p < 0.001$; graph runs: $F(2,67) = 66.23, p < 0.001$). Helmert planned contrasts comparing students in both partial model conditions to those in the control condition revealed no significant difference in the number of bar chart runs, $t(67) = -1.89, p = 0.063, r = 0.22$, but a significantly higher number of graph runs for the partial model conditions, $t(67) = -6.60, p < 0.001, r = 0.63$. Comparison between both partial model conditions further showed that students in the PM+ condition consulted the bar chart tool and the graph tool more often than students in the PM condition (bar chart runs: $t(67) = -3.84, p < 0.001, r = 0.42$; graph runs: $t(67) = -9.82, p < 0.001, r = 0.77$).

Of final interest was whether learning outcomes, performance success, and model testing activities were related. The correlations depicted in Table 2 indicate that students who performed more bar chart runs also performed more graph runs. Both model testing activities were positively associated with students’ model score, meaning that students who ran their model more often also performed more successfully by creating better models. Finally, the number of graph runs and the model score were significantly related to students’ system reasoning knowledge on the posttest.

These statistical analyses demonstrate that support in the form of partial models enhanced students’ domain knowledge, their model testing activities, and the quality of the models they created. Descriptive analyses were performed to shed more light on how partial models shaped students’ interactions with the tool and the task.

*Journal of Research in Science Teaching*
During the model formulation stage, students created an initial model based on their prior knowledge and the information from the instructional text. Students could check the quality of the structure of their initial models in the bar chart. This option was used by all students except four (three in the control condition and one in the PM condition). Analysis of the first model students tried to run with the bar chart showed that these models were largely correct (incorrect elements: $M = 0.08$, Range $= 0–2$; incorrect relations: $M = 0.18$, Range $= 0–4$) but incomplete. Students in the control condition included few variables in their initial model ($M = 2.50$, Range $= 1–4$). Frequently added variables were “blood glucose level,” “glucose release,” and “insulin level”; relations between these variables were virtually absent ($M = 0.05$, Range $= 0–1$). Similarly, students in the PM condition added only two variables on average to the pre-specified model structure ($M = 3.36$, Range $= 2–7$) and rarely connected them by adding relations ($M = 1.04$, Range $= 0–9$). These model expansions were highly divergent and involved all variables from the reference model except “glucose usage fraction.” Students in the PM condition did not have to add variables but they also linked few of the available elements with relations ($M = 0.90$, Range $= 0–5$).

Having completed a model structure, students could run their models with the graph tool. All students in the PM condition used this tool to verify the behavior of their models, whereas only 40% of the eligible students in the control condition ($n = 8$) and 60% of the eligible students in the PM condition ($n = 15$) utilized this feedback option. At this point, taking all conditions into account, students’ models were still largely correct (incorrect elements: $M = 0.20$, Range $= 0–2$; incorrect relations: $M = 0.45$, Range $= 0–5$) but those the students in the control condition and PM condition tried to run were still somewhat incomplete, containing five elements and just over two relations on average (control variables: $M = 5.00$, Range $= 2–7$; control relations: $M = 2.50$, Range $= 0–6$; PM variables: $M = 4.53$, Range $= 2–7$; PM relations: $M = 2.33$, Range $= 0–9$). The models of students in the control condition resembled the ones the PM students inspected with the bar chart tool. All relevant variables except “glucose usage fraction” appeared in some of the models, several local relations between variables were established, and none of the students had specified how the levels of glucose and insulin influence each other. This feedback loop was partly in place in the models PM students ran with the graph tool. Approximately half of these students correctly identified how “blood glucose level” affects “insulin level”—but not the other way around because the variable “glucose usage fraction” was still absent in the students’ models. The PM+ students, by contrast, had greatly expanded their models to include approximately seven variables ($M = 7.10$, Range $= 7–9$) and five relations ($M = 5.76$, Range $= 4–11$). A fully correct feedback loop was found in nearly one third of the models. Running these models and plotting the data in the graph provided insightful information about the behavior of the system they were trying to model. Based on this feedback the PM+ students could fine-tune their models to correctly

Table 2

<table>
<thead>
<tr>
<th>Measure of model testing, performance success, and learning outcomes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Bar chart runs</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>2. Graph runs</td>
<td>0.550&lt;sup&gt;**&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>3. Model score</td>
<td>0.383&lt;sup&gt;**&lt;/sup&gt;</td>
<td>0.693&lt;sup&gt;**&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4. Posttest (system reasoning)</td>
<td>0.111</td>
<td>0.204&lt;sup&gt;**&lt;/sup&gt;</td>
<td>0.428&lt;sup&gt;**&lt;/sup&gt;</td>
<td>—</td>
</tr>
</tbody>
</table>

<sup>**</sup>p < 0.05.
<sup>***</sup>p < 0.01.
describe the system’s behavior, which they actually did until their models were largely correct and complete (see Table 1).

Closer inspection of the PM+ students’ best models revealed no systematic omissions and mistakes. Most students in the PM condition still had not included the “glucose usage fraction” in their models, which prevented them from creating a correct glucose-insulin feedback loop. Local relations between variables were present but often incorrect or incomplete. These inaccuracies did not seem to point to any consistent deviation from the reference model, although it was somewhat remarkable that few students identified how the element “glucose release” determines the increase in “blood glucose level.” The best models created by students in the control condition generally contained the core variables “blood glucose level” and “insulin level” plus three additional correct variables. Here too, “glucose usage fraction” was the most frequently overlooked variable, which inhibited students to correctly model how the insulin level regulates the blood glucose level.

For the model deployment stage, the modelling assignment contained two additional scenarios (eating a pizza, and exploring the effects of Type 1 diabetes) that served to direct students’ activities during the model deployment stage. These scenarios aimed to encourage students to scrutinize their models’ behavior and engage in subsequent testing and debugging activities. The start of the deployment stage was inferred from the log files and defined as the point in time when a variable for the food intake of a pizza was added to the model. Descriptive analysis showed that around 70 percent of the students adapted their models accordingly and hence entered the model deployment stage (control: 78%; n = 18; PM: 65%; n = 17; PM+: 66%; n = 14). Of further interest was how students in the three conditions divided their time between model formulation and model deployment. Students from the control condition, who had to construct their model from scratch, used the least amount of time for model formulation, with an average of 35 minutes (Range 9–57) of their session devoted to model deployment. The PM students spent more time on model formulation and could spend 23 minutes (Range 1–52) on deployment. The PM+ students spent most of their time constructing their model, leaving only 10 minutes (Range 0–33) to work on the additional scenarios in the model deployment stage.

Discussion

This study investigated whether scaffolding by providing partial models promotes the development of students’ domain knowledge, the quality of the models they create, and their model testing activities. The results indicate some clear advantages in all three areas: students who received a partial model checked the accuracy of their model more often as it evolved over time (model testing), created better models (performance success), and evidenced higher gains from pretest to posttest in their ability to reason about the behavior of the glucose-insulin system as a whole (learning outcomes) than students from the unscaffolded control condition. Scores on all three measures were positively correlated and the observed cross-condition differences were more pronounced for students whose partial model was supplemented with an additional list of variables.

The scaffolding mechanisms of structuring and problematizing could help explain these findings. At the outset of this article, two scenarios were proposed to predict the joint effect of these mechanisms on performance success and learning outcomes. The descriptive analyses suggest that the second scenario, which emphasized the role of the problematizing mechanism in enhancing learning outcomes, applies to students in the PM+ condition. They spent most of their time in the model formulation stage trying to create an accurate model. Even though this left them with little time for model deployment, they eventually entered this stage with a reasonably good model and presumably a sound understanding of its content, which appeared to be sufficient to outscore students in the other conditions on measures of performance success and learning.

Journal of Research in Science Teaching
Students in the PM condition entered the model deployment stage sooner, which suggests that the structuring mechanism was more prevalent in this condition. The first scenario predicted that this mechanism would help students reach a certain model quality level faster and thereby increase the opportunities to comprehend the model content in the deployment stage. However, this possibility was contradicted by the fact that performance success and learning outcomes in the PM condition were on par with those of students from the control condition. It thus seems that the PM version of the partial model was insufficiently worked out either to give students a head start in the model formulation stage, as predicted by the first scenario, or to provide them with sufficient hooks to dig deeper into the model structure and understand its content during that stage, as predicted by the second scenario.

Still, the present findings are open to alternative interpretations. For example, one might argue that differences in model quality are an artifact of the students’ instructional condition. That is, partial solutions constrain the space of possible models which, in turn, could reduce the chance of mistakes and hence increase the probability of getting an accurate model. Yet this presumption does not seem to apply to identifying and specifying relations: even if all variables were provided, as was the case in the PM+ condition, students could still define 592 possible relations of which only 14 were correct. Concerning variables, the likelihood of making mistakes was indeed inversely proportional to the number of variables given. Even though normalized model scores were used in the analyses, this does not rule out the fact that identifying relevant variables may be easier when students have fewer options to consider. On the other hand, Mulder et al. (2010) showed that students are actually quite capable of identifying which elements to include in their models, and the descriptive analyses in the present study support this claim. This means that students in the PM condition and control condition could relatively easily increase their normalized model scores, whereas students in the PM+ condition, in contrast, were merely rewarded for the more arduous task of establishing the relations between given model elements. These reasons seem to counter the presumption that the model scores reflect condition constraints rather than student ability.

Another interpretation concerns the possibility of improved judgement accuracy. The performance success data replicate the positive effects of completion problems on performance on computer programming tasks (van Merriënboer, 1990; van Merriënboer & de Croock, 1992) and on concept mapping tasks (Chang et al., 2001, 2002). Baars et al. (2013) recently found that completing partially worked-out examples reduces students’ tendency to overestimate their future test performance more than studying fully worked-out examples does. The extended partial model could have had a similar effect. The descriptive analyses showed that students in the PM+ condition had developed a more complete model structure before they found it appropriate to inspect the behavior of their models over time in a graph. This suggests that the extended partial models helped students evaluate the completeness of their models. As this effect was not yet apparent when students inspected their model structure for the first time with the bar chart, the extended partial models might indeed have improved students’ judgements rather than helped them to initially pick up more information from the text.

These possible differences in judgement accuracy point to yet another reason for the effectiveness of extended partial models. Perhaps students in the PM+ condition devoted their time in the model formulation stage to creating an accurate model, and came to understand its content during model deployment. More specifically, completing the partial solution may have been so demanding—in particular when the extended partial model indeed evoked better judgements—that students had few cognitive resources left for learning. If so, the PM+ students entered the model deployment stage late and with an accurate model of which they did not fully comprehend the behavior. But once freed from the challenge of creating a model, students could
use the little time that remained for this stage entirely for understanding its behavior—which, according to Alessi (2000), could have promoted learning because the model was accurate.

However, students’ learning outcomes seem to support the problematizing option. Results on both knowledge tests indicate that students in all three conditions managed to improve their understanding of glucose-insulin regulation. This result confirms the notion that learning by modelling can be a productive approach to science learning, even in the absence of any kind of instructional support. Still, students who did receive additional scaffolding evidenced higher knowledge gains than students from the control condition. This advantage was only apparent for items addressing system reasoning, and the effects were more pronounced for students who received the extended partial model. So why did students in the PM+ condition learn more about the behavior of the system as a whole? The descriptive analyses show that some students who received the extended partial model already had a fully correct glucose-insulin feedback loop in their model by the time they first consulted the graph tool; others completed this loop shortly afterwards. This suggests that the PM+ students devoted most of their time in the model formulation stage on system reasoning, which might explain their somewhat modest learning of domain concepts and local model reasoning. It thus seems more plausible that students’ system reasoning ability emerged mainly during model formulation. Students in the PM+ condition spent the most time in this stage, and the little time that remained for model deployment seems insufficient to meaningfully explore and learn how the system behaves in various situations. The correlation between the quality of students’ models and their system reasoning scores on the posttest seems to support this notion.

Nevertheless, the basic advantages seen for partial models illustrate that scaffolds based on the example-based learning approach can be successfully applied in learning by modelling. A previous, less successful attempt used heuristic worked-out examples to try to enhance students’ performance success and learning outcomes (Mulder et al., 2014). Although students who received the worked-out examples created better models than their unsupported counterparts, the overall quality of their models was still rather low and the expected improvement in learning outcomes failed to occur. The partial models in the present study represent a different type of worked-out example, and were found to be more effective. The PM+ students in particular managed to create good models and showed substantial knowledge gains on the posttest. This difference in findings might be due to the domain in which the worked-out solution is presented. The heuristic examples in the Mulder et al. study used a topic that differed from the topic of the actual task, which might have decreased their effectiveness. The present study, in contrast, integrated the worked-out example in the actual task and thus the partial solutions and the task addressed the same content. This is actually one of the advantages of completion problems that van Merriënboer (1990) mentions: they offer scaffolding in the relevant domain and thus facilitate translation of the worked example to the task at hand.

Partial models also increased the frequency with which students ran their model with the bar chart and graph tools, and students who received the extended partial model performed these model testing activities even more often than students who received the partial model without the additional set of variables. These results are in line with van Merriënboer and de Croock (1992), who found that partial solutions of a computer program enhanced students’ testing and debugging activities. They emphasized the importance of this finding given the pivotal role of testing activities in programming instruction. Along the same lines, testing activities are key to learning by modelling because they give students the opportunity to verify and refine their initial conceptions of the domain. Given that the students’ initial models were incomplete rather than incorrect, it remains an interesting question how students’ testing and debugging activities are related to model quality. It appeared that students’ debugging activities were mainly geared
toward expanding the model, and PM+ students were more successful in this respect than students from the other conditions. Presumably the extended partial model helped the PM+ students judge the quality (i.e., completeness) of their models, steering them towards more extensive iterative cycles of expanding and verifying their model. The positive correlations between the number of bar chart runs and graph runs and model score seem to support this view. For future research, it would be interesting to collect more qualitative data, for instance, through think-aloud protocols, to further analyze how students expand and verify their model.

To conclude, this study established that partial solutions can enhance students’ modelling performance and learning, and that learners benefit most from partial solutions that provide them with the overall structure of the model and an additional list of variables that could be included in the model. Having established the overall effectiveness of partial models, future research should try to deepen our understanding of the underlying mechanisms that support performance and learning; some specific suggestions were presented in the above paragraphs. Further research is also needed to broaden our understanding of the effectiveness of partial models. Work in this direction might replicate the present findings in other science domains, or verify that learning from partial models has the same benefits over learning from fully constructed models that completion problems have over learning from fully worked-out examples. To further ascertain the effects on performance on other domain problems and the sustained impact over time, future research should include transfer items and delayed posttest measures.

Practical implications pertain to science teachers and curriculum designers responsible for developing modelling tasks for high school students. Learning by modelling can be challenging and sometimes even frustrating, in particular when students have to construct a model from scratch. The present study showed that students benefit greatly from receiving a part of the model they have to create. Designers of modelling tasks are therefore recommended to provide students with a basic model outline and an overview of the elements that can be included in the model. A model outline alone does give students a head start, but is insufficient to maintain and increase the lead over students who create a model from scratch—let alone perform on par with students who receive an additional list of model elements. This recommendation automatically raises the question as to which model elements should be provided. Previous research showed that stocks and flows are difficult for students to grasp. As these elements constitute the basic model structure, it seems advisable to make them available. Which additional variables qualify for revealing cannot be concluded on the basis of the present study because students in the scaffolded conditions either received a complete list of variables or none at all. The frequently neglected variables in the PM condition and control condition further illustrate that this decision relies mainly on the domain and task at hand and should be based on pilot testing. A final design recommendation is to not provide relations other than the flow arrows. Not only would this reduce students’ model formulation activities to a bare minimum, but it also seems redundant because students are perfectly capable of specifying the relations between given model elements.

Science teachers could follow these recommendations in designing modelling tasks for their students. When it comes to the delivery of these tasks, several additional options exist. One would be to have teachers create part of the model in front of the whole class. Such demonstrations would enable teachers to explain their thoughts while constructing the model outline and identifying relevant variables. Although not investigated in the present study, making expert reasoning overt could provide additional scaffolding (e.g., Tabak, 2004). Finally, as creating a model is rarely a goal in itself, teachers should provide opportunities for students to actually use their model, for example, by having them test it in new situations. Model deployment activities such as these could lead to further model improvements and trigger students to examine their models in greater depth, which is essential for meaningful science learning to occur.

*Journal of Research in Science Teaching*
Acknowledgements

This study was conducted in the context of the project “Learning through modelling and self explanations” which is part of the National Initiative Brain and Cognition (NIHC) funded by the Dutch Organization for Scientific Research (NWO), grant no. 056-31-011.

References


Journal of Research in Science Teaching


