Using Co-Lab to build System Dynamics models: Students’ actions and on-line tutorial advice

Crescencio Bravo\textsuperscript{a}, Wouter R. van Joolingen\textsuperscript{b}, Ton de Jong\textsuperscript{b}
\textsuperscript{a}Department of Information Systems and Technologies, School of Computer Science Engineering, University of Castilla - La Mancha
Paseo de la Universidad 4, 13071 Ciudad Real, Spain
\textsuperscript{b}Department of Instructional Technology, Faculty of Behavioral Sciences, University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
Crescencio.Bravo@uclm.es, W.R.vanJoolingen@utwente.nl, A.J.M.deJong@utwente.nl

Abstract
Modeling offers a promising form of constructivist learning for students. By making and executing models of dynamic systems in a computer environment, students are stimulated to learn about the specific domain that is modeled as well as about the process of modeling in general. However, learning by modeling also leads to characteristic student mistakes, based on a combination of faulty domain knowledge and insufficient modeling skills. In this article we describe a method of generating advice to students during their modeling process. The on-line advice system was informed by our observations of a teacher who gave advice via a textual communication tool to students building models with a System Dynamics model editor. The first version of the on-line advice system was evaluated in two ways: first, three teachers evaluated the advice the system generated for students’ final solutions; second, we analyzed the advice the system provided as it was used by a sample of students who were building a physics model. These evaluations showed that the overall approach, including matching a student solution to a family of reference solutions together with the other mechanisms of the advice system, is valid. However, they also highlighted the difficulty of building ‘intelligent’ support to help students to improve their models and gain modeling expertise. The article concludes with a discussion of our current efforts to improve the advice system based on the lessons learnt, which suggest extension of the range of solution representations and of the operations of the advice method.

Keywords. Inquiry Learning, System Dynamics Modeling, Solution Analysis, Intelligent Tutoring Systems
1. Introduction

A basic principle of constructivism is to see learning as an active process in which students develop interpretations from experiences to build meaningful personal knowledge. Computer-based modeling (Penner, 2001) is a promising form of constructivist learning, as students are stimulated by creating and executing models of dynamic systems to learn about the specific domain that is modeled as well as about the process of modeling in general. Within the context of inquiry learning (de Jong, 2006; de Jong & van Joolingen, 1998), modeling can play an important role in expressing the knowledge students gain by doing experiments with a given domain. In a computer-supported collaborative learning environment, modeling helps students to make ideas explicit to their peers when discussing the experimental results (Gijlers & de Jong, 2005; Okada & Simon, 1997).

However, modeling is a challenging activity, primarily due to students’ lack of comprehension of what a model is (Justi & Gilbert, 2000). Spector (2000) indicates that if students are left to their own devices in complex situations, constructions that contribute to meaningful and desired knowledge are not likely to occur. Stratford (1997) assumes that building meaningful models requires substantial prior domain knowledge. Kurtz dos Santos & Ogborn (1994) see the understanding of rates of change as one important difficulty students have when they model. Considering these challenges, Löhner (2005) has compiled conditions under which modeling is feasible for students, which include regular use of a modeling tool and appropriate support or scaffolding. In addition, having specific goals can encourage and facilitate students’ modeling (Hogan & Tomas, 2001).

Unfortunately, in many cases there will be many students and only one teacher in a classroom, meaning that provision of one-on-one support is unfeasible. Therefore, teachers can benefit from a computer colleague that assists them in helping students during their modeling task, playing the role of an intelligent tutoring system. Along this line, this article proposes an automatic method of giving advice to support students during their modeling work. Let’s suppose an inquiry learning setting in which some high school students have been given a theoretical introductory lesson about fluid dynamics, and they must now approach a Water Tank Level (WTL) modeling problem. This problem asks students to create a model to study the water level in a leaking tank, taking into account the influence of the diameter of the tank as well as the size of the hole in the bottom. To do this, they can use a learning environment that offers modeling and experimenting facilities. In this situation, students could benefit from additional support in the form of on-line advice to scaffold their modeling process.

Our goal was to design and implement a flexible and effective advice method that supports domain learning and modeling skill acquisition. We designed such advice support and implemented it in Co-Lab (van Joolingen, de Jong, Lazonder, Savelbergh & Manlove, 2005), a collaborative scientific discovery learning environment in which students, organized in groups, can experiment with simulations and remote laboratories. This allowed us to test, evaluate, and refine the advice method in a real collaborative modeling environment.

We begin by discussing some systems that provide advice in the literature on learning by modeling and by briefly presenting the Co-Lab environment. In Section 3 we describe the development of the method of advice generation, explaining the representation of models and modeling problems and the advice generation procedure. We then report on two empirical studies evaluating the advice generated. Finally, we offer some concluding remarks and outline next steps in this research.
2. Background and related work

Some systems have approached the problems of scaffolding a modeling task by automatically advising or coaching students. DomoSim-TPC (Bravo, Redondo, Ortega & Verdejo, 2006) is a collaborative learning environment for the learning of House Automation (also known as Domotics or Intelligent Building Design). It includes a collaborative simulation tool (Bravo, 2005) incorporating a software assistant that checks the fulfillment of the problem requirements and restrictions in order to inform the students about the quality of their solution. However, this support is not offered as integrated in the modeling process itself, but is shown in a separate workspace. COLER (Constantino-González, Suthers & Escamilla de los Santos, 2003) is a system for collaborative learning of database modeling. It generates advice based on comparing students’ individual and group solutions and on tracking student participation, but it does not take into account model evaluation. CyclePad (Forbus, Everett, Ureel, Brokowski, Baher & Kuehne, 1998) is an intelligent learning environment for engineering thermodynamics. It includes both an e-mail facility to generate help with a design and an on-line coach that gives advice for handling contradictions and adjusting parameters. KERMIT (Suraweera & Mitrovic, 2002) is capable of analyzing student solutions to database modeling problems, using domain knowledge represented as a set of syntactic and semantic constraints. The semantic constraints compare the student’s solution to the ideal one. Belvedere (Toth, Suthers & Weiner, 1997) gives feedback in a collaborative scientific inquiry task. It has two components: an argument coach that advises about the ‘grammar’ of the scientific discourse, and a domain-expert coach that detects whether the students’ joint solution differs from the expert’s solution.

The advice method proposed in this article was targeted for incorporation into the Co-Lab environment. Co-Lab (van Joolingen et al., 2005) is an environment for synchronous collaborative inquiry learning in science domains. The students work in small groups in a shared workspace, doing experiments, making and evaluating models, consulting literature, and discussing their findings. Co-Lab is designed around a building metaphor, in which buildings represent topic domains and floors within a building represent specific content areas of the subject. Students enter each floor in a hall that provides them with an assignment, usually a modeling problem. Students can then move to other rooms. The laboratory contains a phenomenon they can experiment with, in the form of a simulation or an on-line experiment. The theory room allows them to create and execute a model representing the physical system described in the assignment. It contains the Model Editor, which allows students to construct System Dynamics (SD) models and to simulate them, much in the form used by STELLA (Steed, 1992) and Powersim. Finally, in the meeting room students find background literature as well as a tool for planning and reporting on the inquiry process.

Figure 1 shows a Co-Lab session example. The user interface is organized in three main areas. On the left, the tool menu and mechanisms for navigation and coordination are available. At the bottom part, a chat and an object repository can be found. The work area (in the center) houses the tools in use. In the example the group is in the theory room and the Model Editor, the Graph tool and the Table tool are open. When the students call for advice using the check button, a new window is opened showing a list of advice messages, organized as errors, warnings, and comments (an example of this is shown in Section 4.2).
In contrast with the systems mentioned above, the Co-Lab advice method provides interactive fast advice in the modeling workspace, allows the students to select the variable names they want for their model, supports a variety of valid solutions for the problem, separates the solution matching process from the advice generation, and offers wide configuration possibilities. Its design and evaluation are described below.

3. Generating on-line tutorial advice

The following sections present an overview of the advice method proposed in this article as well as its evaluation. Its development followed an iterative and participatory design approach. After designing an initial advice system and performing a pilot study, the design and the advice system itself were refined. Then they were evaluated, first formatively, leading to some recommendations on the system’s configuration and use, and finally summatively. The current section describes the initial design and the pilot study; the following section describes the formative and summative evaluations, in which a number of teachers and students took part.

3.1. The advice system: first design

The initial design of the advice system followed a few distinctive steps. First, students’ solutions to a modeling problem were prepared for comparison with an expert’s solutions. This preparation concerned the analysis of unnecessary intermediate variables and the naming of variables. After that, the students’ models and the expert’s model (the so-called reference solution) were compared and advice was generated based on the outcome of the comparison.

In order to be able to match a student’s model to a given model, both must be represented in the same representational space. We adopted an object-oriented approach in which an SD model is regarded as a set of interrelated objects (stocks, auxiliaries, etc.), which represent variables. The model created by the students needed to be compared to a model that represents a satisfactory solution (the reference solution). However, for many modeling problems multiple equivalent solutions are possible. For instance, a solution may include intermediate variables that may be convenient in representing the model, but that are not absolutely necessary. Two models, one with and one without such variables, may generate identical output. For example, in the WTL problem the outflow rate could be calculated...
directly from the water_volume or an intermediate variable, level in tank, could be used. Our
method needed to allow for such differences, and hence might need to add or eliminate such
intermediate variables from a given model. Also, students may name the variables differently
than in the reference model. This was overcome by associating alternative names with each
variable in the reference solution, in the form of lists of possible words.

After preparing a student’s model in this way, advice generation took place in two phases:
the identification of differences between the reference solution (RS) and the student’s solution
(SS), and the generation of advice based on these differences. Differences between the SS and
the RS were calculated using an algorithm that detected solution components (stocks, flows,
etc.) in the SS that were not in the RS and were therefore unnecessary variables, and RS
variables not included by the student in the SS that were therefore missing variables. “Things”
(units, relationships, etc.) not well defined in the SS were also identified.

Then, an advice message was associated with each specific type of difference. This way,
there was one consistent advice message available for each given difference. Each advice
message had a type (error, warning, or comment) and a level (general or detailed). An error
represented a critical difference between the SS and the RS, generally in relation to the basic
structure of the model (stocks and flows). A warning usually referred to differences in
relation to auxiliaries, constants, connectors, or units. A comment gave a suggestion or
information about the solution. We identified two advice modes: initial and intermediate. As
the students begin modeling, the advice system was in initial mode. When the students had
been working for a specified time or several advice messages had been given, the advice
system changed to intermediate mode. Only general messages were shown in the initial
mode, while detailed messages were shown in intermediate mode.

Furthermore, we wanted to prevent students from using the advice system as a problem
solver instead of creating a solution themselves, by asking for advice on every modeling step.
Therefore, a minimum number of elements in a student’s model had to be present before the
advice system checked the solution. A minimum time also had to pass between checks for
advice to be given. This is compatible with the idea that the teacher (in this case, the advice
system playing his/her role) needs to give students time to build and test their models
(Schecker, 1993).

3.2. Pilot study
The pilot study used a ‘Wizard of Oz’ method (Kelley, 1984) in which a human teacher
mimicked the role of the advice system under design. This method was used to inform the
design with respect to the didactic decisions to be made without being constrained by
technical possibilities. Thus, two students worked individually on a modeling task within Co-
Lab and a remote human teacher gave on-line tutorial advice through a built-in on-line chat
facility.

The task given to the students was modeling the Earth as a black sphere, in which the
main processes are energy flow from the Sun to the Earth and thermal radiation by the Earth
itself. The students and the teacher used the chat for requesting and giving advice
respectively. The teacher had to explain his reasons for giving advice and the students had to
express their reactions to it by speaking aloud, indicating if the advice was understandable,
appropriate, and helpful. We recorded the activity and at the end we gathered the participants’
general opinion about the advice by means of a questionnaire.

The analysis of the audio/video recording together with the chat log showed that at the
beginning, the students built an initial understanding of the modeling problem and of the
model that solves it, and thought in terms of general structures. When their understanding
progressed, they added elements to their general structure, so that they built a more detailed
model and started running simulations in order to complete it. Finally, they refined their
solution by completing small details. Schrader, Lindgren & Sherin (2000) detected the same
behavior when students build system models: students often create their models by thinking first about the elements of the system, and gradually move to identify and incorporate system relationships.

In the questionnaire, the participants were asked whether they thought that the advice could be structured to focus on the objects/elements of the model. The students had conflicting opinions. However, the teacher said:

“I think it is useful, but it is useful in a particular phase.”

He referred to a phase in which the students were developing the conceptual structure and were involved in constructing the model by means of objects.

The teacher indicated that messages of the type “some element is missing/unnecessary in the solution” were suitable. He said:

“This is about what I tried to do: some elements missed, some elements are unnecessary.”

The students were not in agreement again. One of them said,

“Telling me that something is unnecessary or is missing doesn’t make me understand …”

It was clear that giving more general advice messages at the beginning and more detailed ones at the end seemed a good option to the participants. For instance, one student said:

“... at the beginning you have to make up in your mind what the model is about; and while you progress in building the model, you need some help to find out where you have the right stocks and flows.”

In line with the above comments, we arrived at the following principles and ideas to be incorporated in the initial advice approach:

- We identified three types of model structures: initial, in which there is not yet a proper basic structure of stocks and flows present; basic, in which stocks and flows match those in the RS; and complete, which is a basic structure whose auxiliaries, constants, and connectors also match those in the RS. In the WTL problem, the basic structure consists of the water_volume stock and a flow from this stock to a sink. This classification also recognizes an observation in the literature (see, e.g., Mandinach & Cline, 1996): that many students have difficulty analyzing the world in terms of stocks and flows.
- A new stage of the building process was also identified. As a consequence, we added the final advice mode to the two initially considered (initial and intermediate).
- A focus on the model object structure was required for giving advice, but at two levels: in the initial modeling stages (initial mode) the advice should consist of structure-related messages, but without including references to specific objects; in the elaboration and refinement stages (intermediate and final modes), the advice messages should contain explicit object references.
- With respect to advice style, the principle of saying that an object is missing or is unnecessary remained, but we avoided imperative sentences and tried to provoke reasoning and thinking in the students. The indirect style needed to be predominant in the initial modeling stages, but in final stages the direct style could be more useful to guide the students towards a good solution.
- We observed that the teacher did not repeat the same messages when the same differences between SS and RS occurred, but became more precise and concrete each time. To develop this idea, and in harmony with the previous principles, we included structured
variation in the advice. Thus, the message shown for a specific difference could and should vary depending on the context (mode, previous advice messages, etc.).

- We also observed that the teacher gave a maximum of three advice messages each time, always giving the most important one first. This means that we should set a maximum to the number of advice messages generated, and try to position the most important one on top as well.

- We incorporated a number of the advice sentences generated by the teacher. We identified significant sentences such as “This is a nice basic structure for your model,” “You don’t need the <variable>…” or “<variable> depends / does not depend on <variable>.”

According to these principles, we adapted the advice method to contain an extended set of advice messages. The adapted method is described in the next section.

### 3.3. A flexible and ‘intelligent’ advice method

Figure 2 shows the general structure of the redesigned advice method. Two actors are identified: students, who use the Co-Lab Editor to build models and request advice; and teachers, who define solutions to modeling problems and configure the advice method. Three main processes make up the advice system: Matcher, which matches the student’s solution (SS) to the reference solution (RS), generating a list of differences; Advisor, which takes the differences detected and any advice previously given, and generates a list of potential advice messages; and Advice Selector, which sorts and selects the advice messages according to different criteria and then presents them to the students.

Before students can ask for advice, the Editor checks whether the model elements are used and connected according to the SD building rules and whether unspecified elements or relations exist. In this case, the Editor warns the students so that they can correct their model.

The mapping between variable names in SS and RS is done at the beginning of the matching process. The comparison between solutions is similar to the strategy of Belvedere (Toth et al., 1997). This system identifies paths through nodes in the expert’s representation that differ from the corresponding paths in the representation the students are constructing. In our case, we explore the graphs (the variables are nodes and the relationships are arcs) representing the SS and RS, looking for nodes and arcs that do not match.

After one or more differences have been detected, an advice list is generated using the rules in the advice knowledge base (AKB). The AKB contains 44 advice rules. Each rule has
a set of activation conditions to which the differences detected can match. Apart from the type of difference, the activation conditions can also depend on the current advice mode and the history of advice presented earlier. The mode changes depending on the model’s specific structure or on the quantity of student work. When a rule is triggered, its associated advice message is added to the advice message list.

The next step in advice generation is sorting and selecting advice messages. For sorting, we identified five significant ordering attributes for advice messages: (1) the number of times the advice has been given for a specific modeling problem, (2) the advice message type, (3) the type of the main object related to the advice, (4) the name of the main object related to the advice, and (5) the advice level. For each specific modeling problem, any combination of two ordering attributes can be applied. Once sorted, the messages on the advice list can be filtered. There are two complementary possibilities: the Advice Selector can choose the advice messages corresponding to the highest attribute value, rejecting the remaining messages, or it can select the first n messages. Many systems generate only one advice message, e.g., KERMIT (Suraweera & Mitrovic, 2002), BELVEDERE (Suthers, Weiner, Connelly & Paolucci, 1995), and COLER (Constantino-González et al., 2003). However, COLER generates a list of messages and can give the remainder on demand. CyclePad (Forbus et al., 1998) generates several suggestions as advice.

One of the requirements that we outlined for the method was that it should be flexible and easily configurable, so that the teacher could configure the advice method according to his/her preferences and teaching style, resulting in different ways of generating advice, each with different results for the students’ modeling process and for their learning. The configuration parameters are: modeling time and model size constraints for advice request, definition of the change of the advice mode, sorting criteria, and advice message selection rules. This way, the teacher plays the role of creator of the learning activity, because he/she creates the setting and content (modeling problem), and the way the advice system operates.

Table 1 shows a model example for the WTL problem and the advice that was generated for it, within a Co-Lab session. Advice messages are shown for two invocations of the advice system, one in initial and one in final advice mode. The messages were ordered by advice importance and all messages were presented. The student’s model is not correct, and the advice system has given some warnings and comments. The messages in the initial advice mode are more general, trying to make the students think about particular aspects. In the final mode the messages are more specific, guiding the students more directly to a solution. In both cases the last comment encourages the students to simulate the model and compare its behavior with other experimental results because the model’s basic structure (stock-and-flow structure) is correct.

<table>
<thead>
<tr>
<th>Students’ solution</th>
<th>Advice messages in initial mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td>[Warning] Think about what variables (or quantities) flow in or out of the stocks</td>
</tr>
<tr>
<td><img src="image2" alt="Diagram" /></td>
<td>[Warning] Think about the links needed to connect the different variables</td>
</tr>
<tr>
<td><img src="image3" alt="Diagram" /></td>
<td>[Warning] You may consider adding or removing constants (think about constant values)</td>
</tr>
<tr>
<td><img src="image4" alt="Diagram" /></td>
<td>[Comment] This is a nice basic structure for your model! Try running it and compare the results with your experiments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>One ideal solution</th>
<th>Advice messages in final advice mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="Diagram" /></td>
<td>[Warning] Think about whether 'outflow_rate' depends on 'water_volume'</td>
</tr>
<tr>
<td><img src="image6" alt="Diagram" /></td>
<td>[Warning] Use 'm3' to measure 'water_volume'</td>
</tr>
<tr>
<td><img src="image7" alt="Diagram" /></td>
<td>[Warning] Think about whether 'Flow_1' depends on 'outflow_rate'</td>
</tr>
<tr>
<td><img src="image8" alt="Diagram" /></td>
<td>[Warning] You could consider including the constant 'tank_diameter'</td>
</tr>
<tr>
<td><img src="image9" alt="Diagram" /></td>
<td>[Comment] Inserting the optional auxiliary 'level_in_tank' would give you an equivalent solution even more elegant</td>
</tr>
<tr>
<td><img src="image10" alt="Diagram" /></td>
<td>[Comment] This is a nice basic structure for your model! Try running it and compare the results with your experiments</td>
</tr>
</tbody>
</table>

Table 1. An example of a students’ solution and the advice received.
The advice approach presented is valid for a number of educational areas and modeling problems, provided the system state can be expressed with a fixed number of stocks. In this way, the main requirement for applying the advice approach to a modeling problem is the nature of the domain, not the size of the modeling problem.

4. Evaluation studies

The evaluation studies were performed to assess the effects of the advice generated. First, a formative evaluation was performed in a laboratory setting with a few expert teachers. On the basis of this study, a few adaptations were made to the advice system. In the ensuing summative evaluation study, the system was put to the test in a computer classroom with students.

4.1. Method

In both studies, subjects worked on the WTL problem in a classroom as part of their normal class work. In the formative evaluation, twelve Dutch high school students worked on the task without the advice generator being active. Students worked in dyads, creating a total of six models. The models they created were fed into the advice generator for a post-hoc analysis. The resulting advice was assessed independently by three Physics teachers who scored the quality of the mapping between SS and RS variables as well as the advice messages generated. The teachers evaluated the generated advice by filling in a questionnaire using a five-point Likert scale ranging from 1 (totally disagree) to 5 (completely agree). Moreover, they answered three open-ended questions regarding the general advice approach.

In the summative evaluation, twelve Spanish first-year university students of Computer Science worked individually on the WTL problem in a system that did provide on-line advice. Before working on the task, they had a two-hour training session in the use of Co-Lab. The length of the modeling session was set at one hour. The students’ actions, their interactions with the advice system, and the advice messages generated were all logged. After the task, the students responded to a questionnaire concerning the support provided by the advice system.

4.2. Formative evaluation

The first aspect to be validated was the procedure for linking SS variables with RS variables. The system was able to automatically associate 74% of the SS variables with the RS variables. For instance, the list of possible words defined for the RS variable *water_volume* included “water”, “tank”, “water, level”, “tank, level”, “tank, water”, “water, volume” and “tank, volume”; thus, the SS variables *waterTank* or *tank* matched with this list. The remaining variables were linked manually.

The questions presented to the teachers and their average scores are shown in Table 2. We can draw the conclusion that, as judged by the teachers, the advice should have been understandable by the students and could be appropriate for the stage of modeling (initial, intermediate, or final stage). However, the helpfulness of the advice could be improved and the advice seemed to diverge from what the teachers would have said themselves.

<table>
<thead>
<tr>
<th>Question</th>
<th>M</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you think that the advice given is consistent with the solution model and with the reference (ideal) solution?</td>
<td>3.06</td>
<td>0.98</td>
</tr>
<tr>
<td>Do you think that the advice given is understandable by the student?</td>
<td>3.56</td>
<td>0.92</td>
</tr>
<tr>
<td>Do you think that this advice is appropriate for the stage of the solution model?</td>
<td>3.28</td>
<td>0.95</td>
</tr>
<tr>
<td>Do you think that this advice will help the student to improve his/her solution model?</td>
<td>2.83</td>
<td>1.42</td>
</tr>
<tr>
<td>Do you think that the advice is similar to that a teacher would give?</td>
<td>2.72</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 2. Means and standard deviations of the scores of the experts’ questionnaire.
We identified the advice instance that got the worst average score by the teachers. This advice is shown in Figure 3 together with the student’s model. The model that is shown is not very good: it has a disconnected flow and no auxiliaries. In this case, the advice system generated four messages covering too many areas: flows, links, auxiliaries, etc. This is more than a student can assimilate, which could be the reason why the teachers rated this as the worst advice. Specifically, they said that they would remove all the messages but the second one, probably so that the students would focus on the flow of variables in order to fix the disconnected flow.

Moreover, this case highlights a specific problem of the advice system: it seems probable that the students did not know that a flow has to be connected to a stock. In one teacher’s words, “this should be the advice.” The teachers also identified another limitation of the system: the units must match exactly; however, it should be possible, for instance, to work with $m$ instead of $cm$, etc. One teacher thought there should have been a message telling about a flow that is constant (independent of the amount in the stock). This allowed us to detect the flaw that the system did not give advice on relations between variables that the students did not include in their solution. In this case, there was no auxiliary to connect the stock-and-flow structure; therefore the system was not able to say that a link should be inserted, because the auxiliary was missing.

In most cases, the teachers thought some of the advice messages needed to be left out. Regarding message order, they proposed a different order for some messages, but in general they agreed with the order selected.

In the open-ended questions for evaluating the general approach used in generating the advice, the teachers remarked that this approach and the advice style were nice and fine. However, they also agreed it was difficult to give good and helpful advice, because there are different ways of modeling the behavior of a given physical system. One of their opinions was that in the initial phase the students need more help in thinking about the problem, e.g., when the advice system detects that the student is not clear and consistent in modeling a variable (stock). More concrete suggestions were also made concerning the wording of specific messages.

On the basis of the formative evaluation, we made the following adaptations. First, the advice knowledge base (AKB, see Figure 2) was refined: we improved the wording of some advice messages; some new messages were incorporated, e.g., in relation to the generation of advice about RS elements not present in the student’s solution; and some messages (e.g., about units) were moved from intermediate to final mode. Second, new differences (e.g., the case of units not filled) were identified and associated with new advice messages. The ordering of the messages by advice importance appeared to be successful and this was not changed.

4.3. Evaluating the advice system in action

As a starting point, the configuration of the advice generation (see Section 3.3) was set as follows. The first advice generation could take place only after a model with a minimum
number of three elements was present and at least 60 seconds of working time had passed. The advice mode could change from initial to intermediate after a working time of 600 seconds from the start, during which at least three advice requests and four changes in the model were required. Finally, a working time of 1200 seconds in intermediate mode, and at least five advice requests and six changes in the model were required for changing from intermediate to final mode. All this ensured that the students had to do some real work on the model before receiving advice. An alternative possibility was for the model to reach a specific structure: if it had a complete structure the mode was directly set to final, and if this structure was basic the mode was set to intermediate. Because the selection and ordering mechanisms were judged as suitable for advice generation (according to the experts’ opinion in the formative evaluation), we decided to test the advice system again without message selection facilities, in a new modeling problem in a real course situation. The order in which the messages were shown was first by advice message type (error, warning and comment) and then by object name, so that messages related to the same objects were presented together.

During the hour-long modeling session, the 12 students requested advice 102 times and they received 396 advice messages. There were an average of 8.5 checks (SD=2.78) and 33 advice messages (SD=11.64) per student with 3.9 advice messages per check, indicating a very frequent use of the advice system. 83.3% of the students were able to complete the modeling problem.

On average, the first check and call for advice took place after 24 min, 52 s (SD=2 min, 5 s). This represents the time the students needed to read and understand the assignment and to construct an initial model. The remaining checks occurred at an average interval of 3 min, 31 s (SD=4 min, 6 s). Some checks occurred shortly after a previous one, as a result of direct and quick corrections of certain mistakes, and others left more time for reflection and discussion.

The five most frequent differences and advice messages generated (not including advice comments) are shown in tables 3 and 4 respectively. The correct identification of auxiliaries, constants and relation types were the steps causing the most problems to the students in the specific modeling problem proposed. Overall, 196 differences referred to “things” (variables and relations) that were missing, whereas 152 differences referred to “things” that were unnecessary. Because advice messages for a specific difference can vary according to time and depend on the model state, they occurred comparatively less often. The advice aspects seen most frequently were proposal of specific units, insertion of auxiliaries, use of specific types of connectors, and incorrect identification of flows.

<table>
<thead>
<tr>
<th>Difference</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auxiliaries are missing</td>
<td>78</td>
</tr>
<tr>
<td>Constants are missing</td>
<td>65</td>
</tr>
<tr>
<td>Auxiliaries are unnecessary</td>
<td>63</td>
</tr>
<tr>
<td>Constants are unnecessary</td>
<td>46</td>
</tr>
<tr>
<td>A link is not of the correct type</td>
<td>42</td>
</tr>
</tbody>
</table>

Table 3. Highest frequencies of differences.

<table>
<thead>
<tr>
<th>Advice Message</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use a different unit for ‘%1’</td>
<td>19</td>
</tr>
<tr>
<td>Consider adding more auxiliary variables to your model</td>
<td>19</td>
</tr>
<tr>
<td>You may consider adding or removing flows (revise your flow-stock structures)</td>
<td>18</td>
</tr>
<tr>
<td>You do not need all flows you have specified in your model</td>
<td>18</td>
</tr>
<tr>
<td>You should add a connector between ‘%1’ and ‘%2’ of type ‘%3’</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 4. Highest frequencies of advice messages.

With respect to the advice mode, 18.6% of the checks were in initial mode, 29.4% in intermediate mode, and 52% in final mode. Because mode advancement was very restricted
(many working conditions had to be reached), it is clear that the students built good model structures (*complete* structures) that allowed the mode to advance.

We realized that the number of differences decreased as the students advanced and requested advice. From the first check to the seventh check the average number of differences per student decreased as follows: 8.33, 7.59, 7.17, 7.55, 7.27, 6.1 and 4.9. This tendency matched our expectations: the students made fewer mistakes and their models improved over time.

In order to check whether the students actually followed the advice given, we calculated how many models were modified (improved) as result of the advice. On the one hand, we found that 20.9% of the models were not modified from one check to the next, and consequently 79.1% of the models were modified. As an illustration, the two students who did not solve the problem were not able to modify their models in 50% and 33.33% respectively of the cases that they received advice. On the other hand, five students had modified their models in one way or another at every check. But modifying the model does not imply doing it according to the advice. The advice’s degree of success can also be measured by calculating the number of differences (or advice messages) that were repeated from one check to the next, i.e., that were not corrected. Counting advice messages is not useful, because they can vary depending on the advice mode and other conditions. With respect to differences, 66.94% of them (330 differences) were repeated in the next check, which might be regarded as a discouraging result. However, this may be explained by the fact that the students were shown many advice messages at a time, and they were only able to solve a third of them each time (which demonstrates again the necessity of message selection). Therefore, the model modification and difference correction percentages could indicate that the advice is understandable and motivates the students to correct their models.

The answers on the questionnaire revealed that students considered the language used by the advice system in its advice messages to be understandable and clear (M=3.81, SD=0.81), the help they had received to be appropriate to their model and its state, (M=3.67, SD=0.73), the help they had received to be good (M=3.61, SD=0.7), and that the advice system made them work and reflect instead of doing the work for them (M=4.10, SD=0.62). However, they also thought that the advice they had received was not very similar to that a teacher would give (M=3, SD=0.84).

5. **Discussion**

In this article we reported on the design and implementation of a flexible and effective method of advice for scaffolding the complex task of modeling in collaborative inquiry learning environments. The strategy for providing such advice support was based on two main processes: the identification of differences by the comparison of the students’ solutions for a modeling problem with a family of reference solutions, and the generation of on-line advice from an advice knowledge base. This on-line advice system was informed by our observations of a teacher who used a textual communications tool to give advice to students as they were building models with a System Dynamics model editor. The resulting advice system was evaluated in two ways: first, a few expert teachers evaluated the advice generated by the system for some students’ final solutions, and second, we analyzed the advice provided by the system as it was used by a sample of 12 students who were building a model of a Physics system. The first study generated some refinements in the method and the correction of some problems; the second was a summative evaluation.

With this advice-based scaffolding, the students are not left to their own devices in a computer-supported inquiry learning setting: understandable advice is presented that evolves during the modeling process and that is suitable for each situation. In this way the teacher is relieved of providing immediate support and is thus allowed to focus on specific students’ difficulties. However, although the evaluations confirmed that the method is promising, they
also highlighted the difficulty of building an intelligent support for students that helps them improve their solutions and gain modeling expertise. With respect to learning, the studies show some initial evidence that the advice could contribute to the modeling process, as indicated by the improvement of models from one round of advice to the next. More studies with a controlled design are necessary to confirm this and to be able to attribute the improvement to the advice. This would require an experimental set-up with pre- and post-tests about domain knowledge acquired.

The mechanisms we designed to approach the advice problems seem to have been valid: lists of alternative names for the model variables; generation of a family of solutions; identification of different advice modes, model structures and modeling stages; a rule-based knowledge base; ordering and selection of the advice messages, etc. In addition, the configuration and specification facilities of the advice system let the teacher play the role of configurer of the advice system and somehow creator of the learning activity by defining the modeling problem and its solution. Along this line, a further study with different configurations of the method (ordering, selection, change of mode, advice constraints, etc.) would produce interesting results with respect to the most effective configuration for each case and for tuning of the method.

At a more general level, our future efforts will aim at incorporating more intelligence in the advice method. Specifically, an extension of the solution representations is required to cover a family of reference solutions, because the solution to a modeling problem is not unique, and there are different ways of modeling the behavior of a physical system. We will also approach the analysis not only of the model under study but also of the modeling process, so that the advice system can combine knowledge about the students’ thinking and modeling processes with knowledge about the context of the model in order to give the best advice possible. Other extensions of the advice generation method can be to show advice automatically without a request from the students, when the system detects an opportunity to intervene, to generate advice despite the student model’s containing unspecified elements, or to have the advice system be able to perform certain tasks besides just giving messages, e.g., to build model parts.

Acknowledgements
The authors would like to express their gratitude to Jakob Sikken, Ard Lazonder, Sarah Manlove, Petra Hendrikse, Sylvia van Borkulo (University of Twente), Elwin Savelsbergh (Utrecht University), Thorsten Bell (IPN, University of Kiel) and Ernesto Martin (University of Murcia) for supporting this research. The work reported here was done during Crescencio Bravo’s stay at the Department of Instructional Technology of the University of Twente. We would also like to thank the students and teachers of University of Castilla – La Mancha for their participation in the evaluation studies.

References


