International Journal of Artificial Intelligence in Education (2000), 11, 398-415

Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation

CRISTINA CONATI
Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, B.C. Canada V6T 1Z4
email: conati@cs.ubc.ca
Phone: (604) 822-4632
Fax: (604) 822-5485

KURT VANLEHN
Learning Research and Development Center
University of Pittsburgh
3939 O' Hara street.
Pittsburgh, PA, 15260
Email: vanlehn@cs.pitt.edu
Phone: (412) 624-7458
Fax: (412) 624-

We present a computational framework designed to improve learning from examples by supporting self-explanation – the process of clarifying and making more complete to oneself the solution of an example. The framework is innovative in two ways. First, it represents the first attempt to provide computer support to example studying instead of problem solving. Second, it explicitly coaches a domain-general, meta-cognitive skill that many studies in cognitive science have shown to greatly improve learning.

The framework includes solutions to three main problems: (1) to design an interface that effectively monitors and supports self-explanation; (2) to devise a student model that allows the assessment of example understanding from reading and self-explanation actions; (3) to effectively elicit further self-explanation that improves student's example understanding. In this paper, we describe how these solutions have been implemented in a computer tutor that coaches self-explanation within Andes, a tutoring system for Newtonian physics. We also present the results of a formal study to evaluate the usability and effectiveness of the system. Finally, we discuss some hypotheses to explain the obtained results, based on the analysis of the data collected during the study.

INTRODUCTION

Research on Intelligent Tutoring Systems (ITS) has been increasingly affecting education. While for many years ITS remained confined to research labs, today they have started moving into the classroom, showing their effectiveness for learning and influencing the structure of traditional curricula (Koedinger, Anderson, H., & Mark, 1995). However, existing ITS still target only a limited part of the learning process. They generally focus on teaching problem solving and domain specific cognitive skills.

The long-term goal of our research is to explore innovative ways in which computers can enhance education by covering other learning phases and by helping students acquire meta-cognitive, domain independent learning skills. In this paper, we describe our first step in this direction: a computational framework designed to support learning from examples and the meta-cognitive skill known as self-explanation – generating explanations to oneself to clarify an example's worked out solution.

Effectively learning from examples is important because students heavily rely on examples when learning a new skill (Anderson et al. 1981, Pirolli & Anderson 1985, LeFevre & Dixon 1986, VanLehn 1986). However, the benefits of learning from examples strongly depends on how students study them. Several studies in cognitive science show that students who spontaneously self-explain when they study examples learn more (Chi, Bassok, Lewis,

Reimann, & Glaser, 1989; Ferguson-Hessler & Jong, 1990; Pirolli & Recker, 1994; Renkl, 1997; Renkl, Stark, Gruber, & Mandl, 1998). Furthermore, self-explanations are usually more effective than explanations provided by others, because (Chi, in press)

- they trigger more constructive learning processes, by requiring students to bring to bear and actively elaborate their existing knowledge, and
- students initiate self-explanations to address their specific problems in understanding the example, while often external explanations are not tailored to a student's individual needs.

The self-explanation studies also show that most students do not spontaneously self-explain. However, students start self-explaining more when they are guided (Bielaczyc, Pirolli, & Brown, 1995; Ryan, 1996) or even just prompted to do so (Chi, Leeuw, Chiu, & LaVancher, 1994). These results suggest that it can be greatly beneficial to integrate computer-based support to problem solving with individualized guidance to learning from examples through self-explanation.

To provide this guidance, a computer tutor must be able to monitor students as they study examples and to elicit further self-explanation that can improve the students' understanding. Two tasks apparently simple, but that entail additional challenges for the traditional ITS problems: user interface design, student modelling and providing adequate help.

- User interface design. In previous studies, example studying and self-explanation consisted of reading and speaking. How can we devise an interface that monitors students' attention and allows them to constructively generate their self-explanations, given that eye-tracking technology and natural language processing are still not powerful and reliable enough to be readily usable in non-laboratory setting?
- Student modelling. To model a student during example studying requires assessing how well the student understands the example and learns from it. How can we perform this assessment by relying on actions like reading and self-explaining, that are largely ambiguous and have less direct correspondence to example understanding than problem solving actions have to problem solutions?
- Providing adequate help. One of the benefits of self-explanation comes from the fact that spontaneous self-explainers selectively generate self-explanations to target their specific learning needs. How can a computer tutor decide what further self-explanations can be more beneficial for those students that do not spontaneously self-explain? When and how should the tutor elicit these self-explanations from those students that are naturally reluctant to self-explain?

Our framework to support self-explanation, know as the SE (Self-Explanation)-Coach, includes solutions to these problems. The solutions are grounded in existing hypotheses of what are the salient features that make self-explanation effective for learning and are the result of a thorough process of iterative design. The framework has been implemented and tested within Andes, a tutoring system for Newtonian physics that supports students during both example studying and problem solving (VanLehn, 1996). During example studying, the SE-Coach makes sure that students thoroughly self-explain the available examples, especially those parts of the solutions that may be challenging and novel to them. Figure 1 shows one of the SE-Coach examples, which reflects the structure of most examples presented in physics textbooks.

The paper is structured as follows. After discussing related work, we describe the cognitive science findings that provide the theoretical justification for the SE-Coach's design. Then, we illustrate the SE-Coach's architecture and the knowledge representation underlying the system's expertise on self-explanation. Next, we describe the menu-based interface that monitors students' attention and provides structured prompting and scaffolding for self-explanation. We then give an brief overview of the SE-Coach's student model, based on the probabilistic reasoning framework of Bayesian network (Pearl, 1988) and we illustrate how the SE-Coach uses the model to elicit further self-explanations that improve example understanding. Finally, we discuss the results of a formal study that we performed to evaluate the SE-Coach usability and effectiveness for learning.

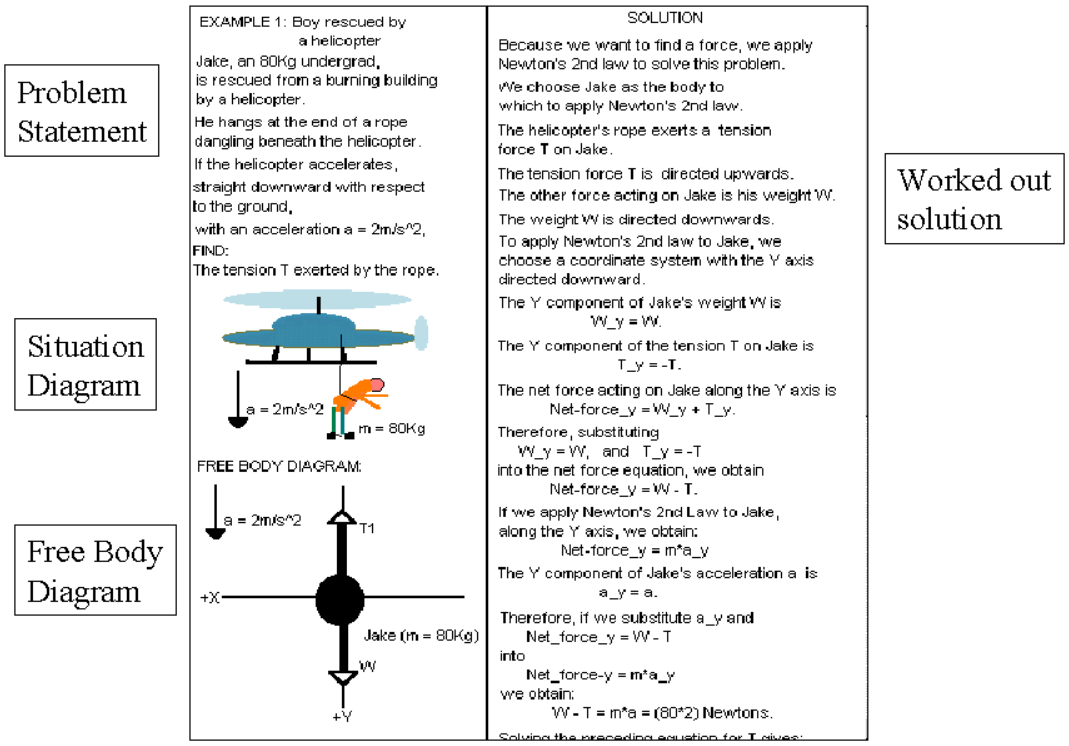

Figure 1: A sample phisics example and its components

RELATED WORK

Using explanations to enhance learning has been a prominent research topic in the ITS community. Most of the work on this subject has focused on how to enable a computer tutor to generate explanations that can facilitate the students' learning (Clancey, 1990; Moore, 1996; Moore, Lemaire, & Rosenblum, 1996; Vivet, 1987; Woltz, McKeown, & Kaiser, 1990). More recently, researchers have started investigating computer tools that support learning by facilitating the exchange of explanations among peers (Baker, 1999; Ploetzner & Fehse, 1998). Although this research, like ours, aims to trigger learning by bringing the students to generate active explanations on the target instructional material, it focuses on a different instructional setting: learning through collaboration with peers. In contrast, the work presented in this paper focuses on how to help students generate and learn from explanations when there are no peers to stimulate and validate the process. Supporting self-explanation is important not only because learning with peers is not always feasible, but also because self-explanation has two main pedagogical differences from generating explanations for others. First, when self-explaining, learners can target specific problems in their understanding rather then having to concentrate on what is unclear to someone else. Second, when self-explaining, learners do not need to worry about phrasing the explanations properly, and can therefore concentrate on the explanation content. These differences do not imply that self-explanation is better than explaining to others. They simply contribute to make self-explanation a different meta-cognitive skill, that can be useful at different learning stages and that improve a student's general learning ability.

Like the SE-Coach, other tutoring systems rely on examples as instructional means. However, most of these systems use examples to support students as they solve problems, not as a specific learning phase prior and complementary to problem solving. ELM-PE (Burrow & Weber, 1996) and ELM-ART (Weber & Specht, 1997), allow the student to access relevant examples while solving LISP programming problems and provide explanations on how each example is relevant for the problem solution. SHERLOCK (Gott, Lesgold, & Kane, 1996),

provides expert solutions to troubleshooting problems, and helps students compare these solutions with their own solutions at the end of each problem solving task. CATO (Aleven & Ashley, 1997) helps students building legal arguments by generating relevant example cases and by reifying the connection between the content of the cases and their use in the arguments. Besides using examples in a different instructional situation, none of these systems tries to encourage students to view the examples, nor do they monitor how students study and understand them. Moreover, the systems themselves, rather than the students, generate elaborations on the presented examples.

The Geometry Tutor (Aleven, Koedinger, & Cross, 1999) moves a step closer to the SE-Coach. It explicitly encourages students to explain, in term of geometry axioms, the problem solving steps they have used to build a geometry proof. However, there are three main differences with the SE-Coach. First, the explanations are generated during problem solving. Second, an explanation with the Geometry Tutor consists simply of a selection from a list of geometry axioms. The student does not have to explain the axiom any further. Third, the tutor makes the students explain each solution step. It does not take into account the students' knowledge or previous interactions with the system to evaluate if some explanations may be more beneficial than others for the students.

PRINCIPLES UNDERLYING THE SE-COACH'S DESIGN

Incremental support to self-explanation

Our framework for self-explanation is designed to provide incremental support to self-explanation through different levels of prompting and scaffolding, embedded in the interface design and in the SE-Coach tutorial interventions. These different levels aim to help students with different self-explanation capabilities self-explain more, while maintaining as much as possible the spontaneous, constructive nature of this learning activity.

On the one hand, there are arguments for giving students the initiative during self-explanation. Self-explanation enables students to question and repair their understanding (Chi, in press), in ways that can be different across students and learning situations (Renkl, 1997). Moreover, students with good self-explanation and self-monitoring skills often can repair what they do not understand better than teachers can, because teachers generally cannot diagnose as precisely the students' comprehension problems. (Webb, 1989). These arguments call for an interface that leaves much of the initiative to the student.

On the other hand, there are arguments for giving the initiative to the Coach. Self-explanation studies show that many students do not self-explain, for a variety of reasons.

1. Many students are not good at self-monitoring when they study (Chi, in press; Chi et al., 1989). They tend to overestimate their understanding of examples (Renkl, 1997) and therefore do not initiate self-explanations to improve it.
2. Sometimes students are unable to use their domain or common-sense knowledge to generate meaningful self-explanations. This mainly happens when the examples are complex enough that most students are aware of having comprehension gaps, but still cannot generate self-explanations that repair these gaps (Renkl, 1997).
3. Even students that spontaneously engage in self-explanation do not always generate the kinds of explanations that are most useful for learning. For instance, explanations that relate steps in the example solution to goals in the underlying solution plan generally help learn highly transferable knowledge (Catrambone, 1995; Pirolli & Recker, 1991; Renkl, 1997; Renkl et al., 1998). However, even spontaneous self-explainers tend to generate few goal-oriented self-explanations (Renkl, 1997).

The different levels of prompting and scaffolding in the SE-Coach are designed to accommodate the varied propensity and capability to self-explain that different students have, so as to provide each student with the minimum intervention sufficient to trigger constructive and effective self-explanations.

Focus on correct self-explanation

Our self-explanation framework includes the capability of providing feedback for correctness on the students' self-explanations. The issue of whether such feedback should be provided is controversial. In all the experiments on self-explanation, any statement that went beyond the information presented in the worked out solution was classified as self-explanation, be it correct or not. In the experiments in which human tutors guided self-explanation, the experimenters elicited additional clarifications from the students when their self-explanations were incomplete or incoherent, but did not give feedback on their correctness (Bielaczyc et al., 1995; Chi et al., 1994). In all these experiments, students' problem solving improved, leading some researches to argue that is the self-explanation process per se, and not the correctness of its outcome, that elicits learning (Chi, in press; Ryan, 1996). In particular, Chi (Chi, in press) argues that incorrect self-explanations are beneficial exactly because they create flaws in the student's knowledge. These flaws may be later contradicted by other elements of the example, triggering self-explanations to fix the flaws and thus generating better learning.

However, this argument applies only to students that can monitor their understanding and we know that these students are a minority. The other students may seldom detect the inconsistencies generated by their incorrect self-explanations. Immediate feedback on self-explanation correctness protects these students from learning wrong knowledge from incorrect self-explanation, and simply makes the other students detect the conflict sooner than they would on their own. Thus, although we believe that even incorrect and incomplete self-explanations can improve learning, we agree with (Renkl et al., 1998) that helping students generate more correct self-explanation can extend these benefits.

Focus on domain-based self- explanation

In order to provide feedback for correctness, the SE-Coach needs to have an internal representation of the relevant, correct self-explanations that can be generated for each available example. It would be unfeasible to encode these explanations by hand, especially because we ultimately want to allow instructors to easily extend the set of available examples on their own. Thus, we identified in the literature two types of self-explanations that can be automatically formalised in a computational model, given a rule based representation of the underlying domain knowledge. These types are:

a) Justifying a solution step in terms of the domain theory, and

b) Relating solution steps to goals in the abstract plan underlying the example solution.

These self-explanations have been shown to highly correlate with learning and were common across the different instructional domains investigated in the self-explanation experiments (physics, statistics, programming, physiology of the human circulatory systems). We label these self-explanations "domain-based" because they involve relating example steps to the target domain knowledge, as opposed to self-explanations that involve common sense knowledge. Common-sense based self-explanations also seem to play an important role in learning from examples (Chi & VanLehn, 1991; Chi et al., 1994; Bielaczyc et al., 1995; Ryan, 1996). Currently, the SE-Coach cannot support these self-explanations, because doing so would require a natural language interface and much more complex domain and student models. However, even if the SE-Coach cannot explicitly guide common-sense based explanations, hopefully it does not prevent the students from generating them spontaneously.

Principled design of interface tools to support self-explanation

After identifying the kinds of self-explanations that the SE-Coach could support, we had to understand which interface could help students generate them. Since using natural language input was unfeasible, we needed menu-based tools that could still allow the students to generate their self-explanations as naturally and constructively as possible. (Chi et al., 1989) analysed the form of students' spontaneous self-explanations and identified two forms that were highly used: (1) expanding or refining the preconditions of a solution step and (2) explicating and inferring

additional consequences of a step. Hence, we have designed menu-based tools that scaffold self-explanations with these forms. The SE-Coach's menu-based tools allow students to justify a solution step by describing the domain rule from which the step derives, in terms of (i) the preconditions that must be verified to apply the rule; (ii) the results that the rule application generates. This description in terms of preconditions and consequences reflects the SE-Coach's rule-based domain representation, and allows the system to provide feedback for correctness based on this representation.

A student model to guide the SE-Coach interventions

The self-explanation interface is a scaffolding tool meant to encourage students to spontaneously self-explain. However, a framework to support self-explanation must be able to provide stronger interventions to help those students who are not receptive to the interface scaffolding.

One way to provide this stronger scaffolding could be to make the students use the interface tools to self-explain every example part. This strategy may possibly work with students that never self-explain, but would end up suggesting redundant self-explanations to the others. This could have negative influence on the students' motivation and trust in the system's effectiveness, reducing the likelihood that students would follow the SE-Coach's suggestions. Therefore, it is particularly important that the SE-Coach generates tutorial interventions that the student can perceive as relevant and useful for learning.

The SE-Coach's student model is designed to assess when students are spontaneously self-explaining without using the interface tools, in order to avoid burdening students with requests of self-explanations that they have already generated. It also assesses self-explanations generated through the interface tools and uses its assessment to detect gaps in the student's example understanding. The SE-Coach focuses its interventions on eliciting further self-explanation that fills these gaps, as students that are natural self-explainers do.

THE SE-COACH'S ARCHITECTURE

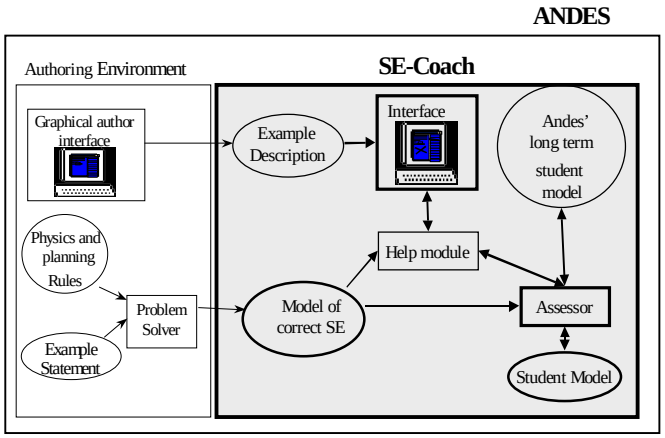

Figure 2: SE-Coach's architecture

As we mentioned in the introduction, the SE-Coach has been implemented within the Andes tutoring system for physics (VanLehn, 1996). Figure 2 shows the SE-Coach's architecture. Prior to run time (left side of Figure 2), an author creates both the graphical description of the example and the corresponding coded definition of the example statement. A Problem Solver uses this definition and the set of physics and planning rules representing Andes' domain knowledge to automatically generate a model of correct self-explanation for the example solution. The model is a dependency network that encodes explanations in terms of how intermediate goals and facts in the example solution are derived from physics and planning rules.

At run-time, students use the SE-Coach's interface to interactively study examples and generate self-explanations. The interface sends the student's explanations to the Help module, which tries to match them with elements of the self-explanation model and provides immediate feedback as to whether the explanations are correct or incorrect. The student's interface actions are also sent to the Assessor module, that uses them to update the SE-Coach's student model. The student model is a Bayesian network (Pearl, 1988) that integrates information on (i) the student's actions, (ii) the model of correct self-explanation and (iii) the student's domain knowledge (encoded in Andes' long term student model) to assess the student's understanding of the example. The SE-Coach's help refers to the student model to make decisions about what further self-explanations to elicit from the student.

THE MODEL OF CORRECT SELF-EXPLANATION

The model of correct self-explanation (SE model from now on) is the core structure of the SE-Coach's expertise. This model encodes the knowledge to provide feedback on student's self-explanations, it is used in the student model to assess how the students' self-explanations reflect example understanding and it guides the SE-Coach's tutorial interventions.

A Problem Solver automatically generates an SE model for each new example added to the SE-Coach's set, starting from Andes rules and from a coded definition of the example problem statement (see Figure 2).

R-body-by-force
If there is a goal to select a body for Newton's 2nd law
and the problem goal is to find a force on an object
then select as body the object to which the force is applied

R-try-Newton-2law
If the problem goal is to find a force
then set the goal to try Newton second Law to solve the problem

R-goal-choose-body
If there is a goal to try Newton's second law
then set the goal to select a body to which to apply the law

Figure 3: sample rules in the Andes' knowledge base.

Andes' rules are based on the representation used by Cascade (VanLehn, Jones, & Chi, 1992), a cognitive model of learning through self-explanation of Newtonian physics examples. Andes' rules have been developed in collaboration with three physics professors at the U.S. Naval Academy, the domain experts for the Andes project. The rules represent qualitative and quantitative physics knowledge sufficient to solve Newtonian physics problems (see Figure 3, top box, for a sample physics rule). They also represent planning knowledge encoding the abstract planning steps that an expert might use to solve these problems (see Figure 3, center and bottom box, for sample planning rules). Thus, the Problem Solver produces a hierarchical dependency network that encodes how an example solution's qualitative results and equations derive from physics rules, along with the abstract plan underlying the solution.

To generate the SE model, the Problem Solver starts with (i) a set of facts describing the example initial situation; (ii) one or more goal statements that identify the example sought quantities. From the initial set of facts and goals, the Problem Solver begins applying rules in the knowledge base, generating new sub-goals and facts until it finds all the unknown quantities necessary to compute the example sought quantities. For instance, consider the segment of SE model shown in Figure 4, which relates to the example in Figure 1. The Problem Solver starts with the top-level goal of finding the value of the force on Jake (node *G-force-on-Jake* in Figure 4). From this, it applies the rule *R-try-newton-2law* (defined in Figure 3) and forms the

sub-goal of using Newton's second law to find the desired force (node *G-try-newton-2law* in Figure 4).

Next, it applies the two rules *R-goal-choose-body* (shown Figure 3) and *R-find-forces* to generate the two sub-goals *G-goal-choose-body* and *G-find-forces*, corresponding to two first level goals in the plan to apply Newton's second law. When the problem solving terminates, the outcome is a partially ordered network of goals and intermediate results (or facts) leading from the top-level goal to a set of equations that are sufficient to solve for the sought quantity, the magnitude of the force on Jake. Figure 4 shows the section of the SE model up to the application of the rule *R-body-by-force*, that selects Jake as the body to which apply Newton's law (node *F-Jake-is-the-body*) and of the rule *R-tension exists*, that identifies the existance of a tension force on Jake (node *F-tension-on-Jake*).

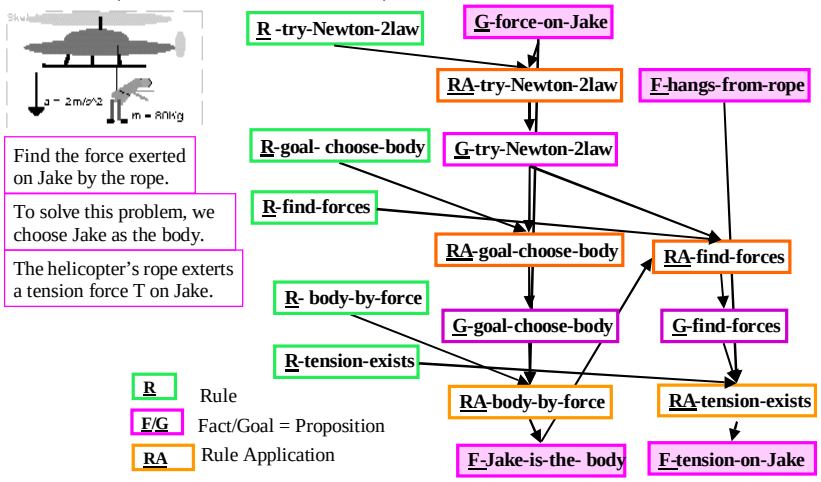

Figure 4: segment of SE model for the partial example solution on the left

Every element of the example statement, worked-out solution and graphics corresponds to a fact or goal (proposition) node in the SE model (proposition nodes are labelled with a "F-" or "G-" prefix in Figure 4). However, the SE model can contain proposition nodes that do not correspond to any element in the example text, if the example leaves out some details of the solution derivation. For instance, only the shaded proposition nodes in Figure 4 (*G-force-on-Jake*, *F-hangs-from-rope*, *F-Jake-is-the-body* and *F-tension-on-Jake*) correspond to steps explicitly expressed in the example solution shown on the left of the figure.

Links in the dependency network encode how each proposition node derives from physics and planning rules (nodes labelled with the "R-" prefix in Figure 4) and from the proposition nodes that match the rules' preconditions. Derivations of proposition nodes from rules and other propositions are explicitly encoded in the network by rule-application nodes (labelled with the "RA-" prefix in Figure 4). These derivations correspond exactly to the explanations that the SE-Coach targets:

- How solution steps can be justified in term of physics principles.
- What goal each solution step accomplishes in the plan underlying the example solution.

Hence, the dependency network provides a model of correct self-explanation that the SE-Coach can use to evaluate the student's explanations and to decide what further explanations can improve the student's understanding. The model is similar in nature to the rule-based domain (or expert) models that other intelligent tutoring systems use to support students during problem solving (Anderson, Corbett, Koedinger, & Pelletier, 1995; Clancey, 1990; Gertner, Conati, & VanLehn, 1998). However, because the model explicitly distinguishes between rules encoding domain knowledge and rules encoding planning knowledge, and because it explicitly represents the application of these rules as nodes in the dependency network , the model is especially suitable to represent and monitor the generation of the self-explanations that the SE-Coach targets, as we will see in the next sections.

THE SE-COACH'S INTERFACE

As we discussed earlier, the SE-Coach provides incremental support to self-explanation through different levels of scaffolding. Three of these levels are embedded in the interface design, described in this section. A fourth level is provided by the SE-Coach's tutorial interventions, as we illustrate in the next section.

Attention monitoring and control

The first level of scaffolding in the SE-Coach's interface is provided by a masking mechanism that presents different parts of the example covered by grey boxes, each corresponding to a "unit" of information (see Figure 5). When the student moves the mouse over a box, it disappears, revealing the text or graphics under it. While reading a line in the textual part of the example solution, the student can refer back to the situation diagram or to the free body diagram by clicking on the left or right mouse button respectively. Figure 5 shows how the example in Figure 1 looks in the masking interface, when the student moves the mouse over the second solution line and clicks to uncover the free body diagram.

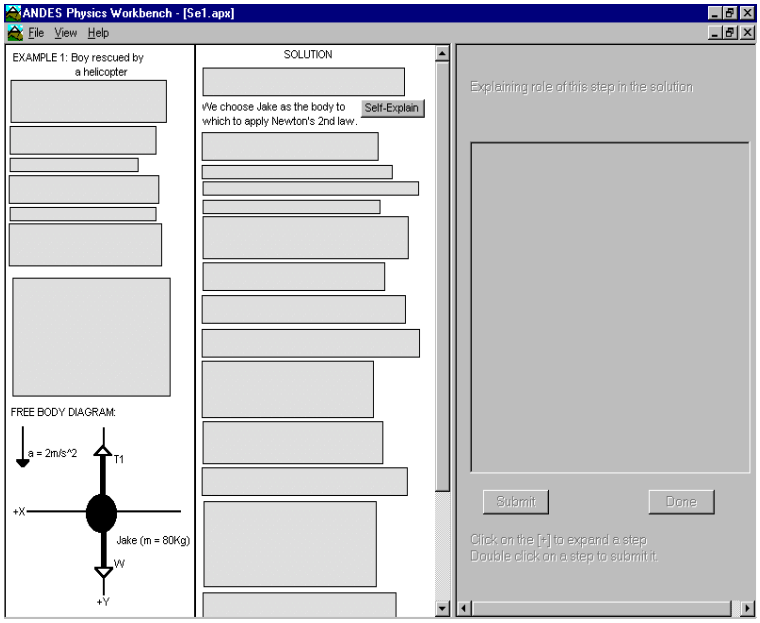

Figure 5: example in Figure 1 presented with the masking interface

The masking interface allows the SE-Coach to track what the student is looking at, and for how long. When a new example is added to the SE-Coach, each item in the masking interface is linked to the corresponding proposition node in the SE model. These links allow the SE-Coach to attach information on student attention directly onto components of the SE model. This information is one of the parameters the SE Coach uses to assess whether and what the student is self-explaining, as we will see in the next session. The mapping between interface items and SE model nodes can be many-to-one, because different items can encode the same fact or goal (like, for instance, the Free Body Diagram object labelled as "*Jake*" and the uncovered line "*we choose Jake as the body…*" in Figure 5). The many-to-one mapping allows the SE-Coach to recognise example parts that call for the same explanation and to avoid eliciting redundant explanations from the students.

The masking interface is a relatively simple way to obtain information on students' attention, without using complex eye tracking devices. However, we were afraid that this unusual way of presenting examples could interfere with reading and understanding them. Thus, we performed a pilot evaluation of the masking interface at an early stage of the system design. Besides

verifying the usability of the masking interface, the evaluation aimed to compare it with an alternative design in which the example parts were faded instead of masked

Ten first year college students studied one example with the masking interface and a second example with the faded interface. We alternated what interface we showed first, to avoid presentation bias. Then, we asked the students to choose between the two interfaces to study a third example. Six students chose the faded interface and four the masked interface, but almost all students found the two interfaces fairly equivalent. Furthermore, not only were none of the students annoyed by the fact the example parts had to be explicitly uncovered. Most students said that both interfaces helped them study the examples more carefully, because they encouraged them to focus on a single item at a time. Thus, in addition to tracking students' attention, the masking interface provides a first level of scaffolding to increase students' self-explanation.

Since the students did not seem to strongly prefer one interface over the other, we kept the masking one, because it is quite difficult to adjust the faded interface so that a student can identify the different parts of an example without reading their content. Furthermore, the right contrast changes in monitors with different resolution, making it impossible to find a setting that works in all situations. We continued to probe the students' attitude toward the masking interface throughout the successive evaluations of the system, which confirmed the students' positive reaction to the interface. The only criticism that some students had concerned not being able to see more than one line at a time when reading long algebraic derivations. To fix this problem, we modified the interface so that students can uncover up to three lines at a time, by pressing the control key as they move the mouse over the lines that they want to see at once.

Prompts to self-explain

The second level of scaffolding is provided by the SE-Coach's interface through specific prompts to self-explain. Whenever the student unmasks a piece of the example, if it contains an idea worthy of explanation the interface will append a button labelled "self-explain". Pressing the button produces simple prompts to initiate self-explanations in terms of domain principles (e.g., *"this choice is correct because…"*) and abstract solution plan (e.g., *"the role of this choice in the solution plan is to…"*).

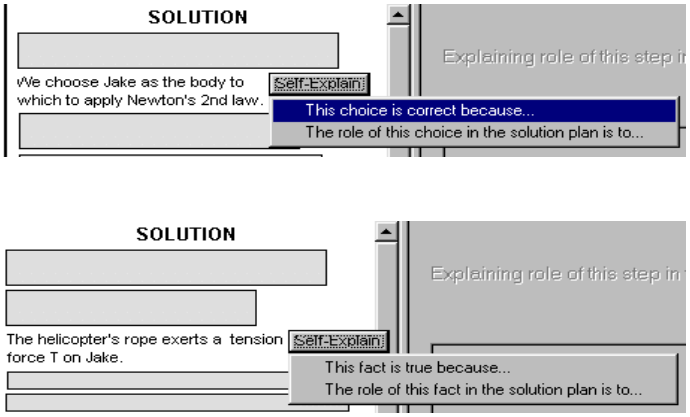

Figure 6: SE-Coach prompts to self-explain

Figure 6 shows the different prompts associated with the second and third line of the example solution in Figure 5. These prompts are designed to elicit self-explanation by stimulating self-questioning (Webb, 1989). Self-questioning seems to be an effective approach to counteract the students' tendency to overestimate their example understanding, because it leads the students to (a) ask themselves questions that target important but possibly problematic knowledge about the example; (b) initiate self-explanation actions if they cannot answer these questions (Chi, in press; Webb, 1989). In particular, by having prompts that trigger plan-related questions, the SE-Coach should stimulate students to generate a type of self-explanation that, as we discussed earlier, is especially unnatural to them but extremely useful for learning.

Interface tools to generate self-explanations

The third level of scaffolding in the SE-Coach interface consists of menu-based tools that provide constructive but controllable ways to generate the desired self-explanations. These tools aim to help those students that would otherwise be unable to self-explain effectively, even if they realize their need to do so (Renkl, 1997).

If a student selects the first choice in the prompting menus shown in Figure 6, a Rule Browser is displayed in the right column of the window (see Figure 7). If the student selects the second choice in the prompting menus, then a Plan Browser is displayed (see Figure 10). The next subsections describe how the interaction proceeds in the two cases.

The Rule Browser

The Rule Browser (see Figure 7) contains all the system's physics rules, organized in a tree structure similar to the Windows file system, so that clicking on the + and – buttons reveals and hides subtrees of the hierarchy. Using this familiar interface, the student finds and selects a rule that she thinks justifies the currently uncovered example item.

If the student then presses the "submit" button at the bottom of the Browser, the SE-Coach will provide feedback to indicate whether the selected rule is the one that explains the uncovered unit of information.

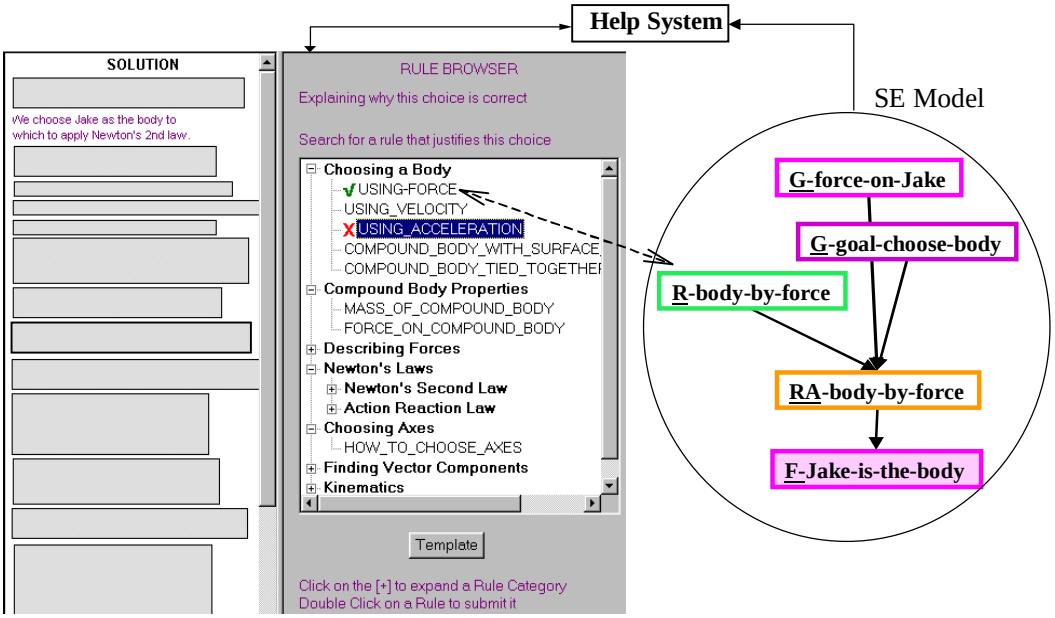

Figure 7: The Rule Browser

Consider the situation in Figure 7, in which the student activates the Rule Browser to self-explain the second solution line and selects the correct rule (marked with a checkmark) after selecting an incorrect one (marked with a cross). To provide feedback on the student's selection, the SE-Coach:

1) retrieves in the SE model for the current example the proposition node corresponding to the uncovered example part (in this case, the shaded proposition node in Figure 7).

2) Finds the closest rule node among the ancestors of the proposition node (the node *R-body-by-force* in Figure 7) and checks if it corresponds to the rule that the student selected.

A green checkmark will appear beside the Rule Browser selection if the rule is correct, a red cross will appear otherwise (see Figure 7).

The SE-Coach does not provide additional help besides red/green feedback. This avoids interfering with what is considered the key factor that makes self-explanation effective for learning: that to generate their self-explanations the students elaborate the available material and knowledge by themselves (Chi, in press). Thus, when a wrong rule is selected, the only way for the student to correct the mistake is to keep browsing the hierarchy until the correct rule is found. For this reason, the organization of rule names in the Browser is crucial to make the search for the correct rule a thought provoking activity, instead of a frustrating one that may result in the student clicking exhaustively on all the entries.

The current organization of the rule hierarchy is the result of successive evaluations with pilot subjects, which helped reduce the amount of floundering observed in the first versions of the Browser. A quite interesting behavior that surfaced during these evaluations is that most students did not try to click on rule names randomly when they got stuck. Rather, when they could not find plausible candidates in the category that they had expanded they would stop, without even trying to browse other parts of the hierarchy. We repeatedly changed the categories' names and arrangement to maximize the chance that students immediately enter the right part of the hierarchy. We also provided cross-references for rules that could plausibly belong to different categories, such as the rule encoding the definition of Net Force, which rightfully belongs to the category *Newton's Second Law*, but that students often tried to find in the category *Describing Forces* (see Figure 7).

Feedback from the pilot evaluations suggested another important modification to the original interface design: the possibility to go back and browse through the example while using the Rule Browser (or any other tool for self-explanation). In the original design, this was not allowed, and many pilot subjects complained that they needed to review the example solution in order to complete their self-explanations with an interface tool. In the current interface, while any of the interface tools is open, the student can still uncover other parts of the example, beside the one for which self-explanation was initiated. The part that is currently explained turns pink, to remind the student of what is the current focus of self-explanation.

The Rule Templates

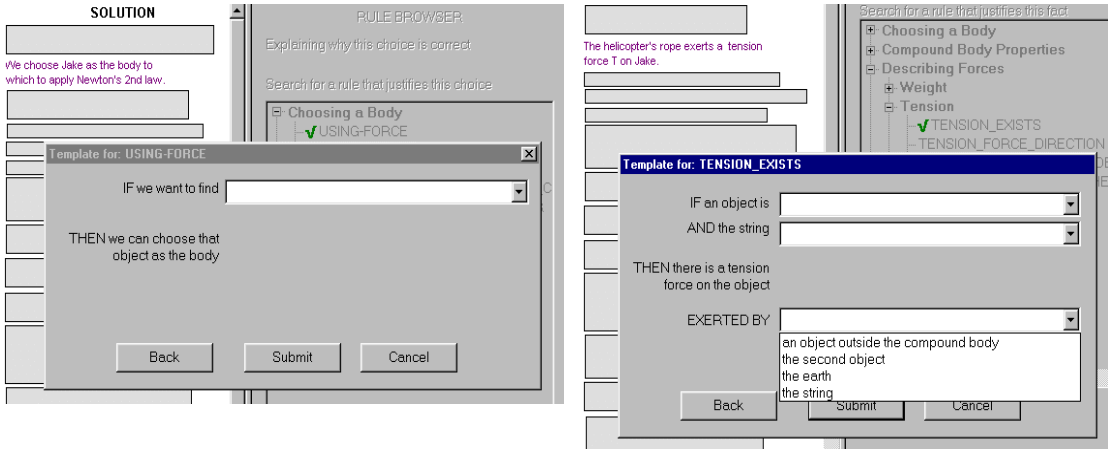

Figure 8: Rule Templates

The Rule Browser lists only rule names, and most students will need to know more about a rule before they can be sure that it is the explanation they want. To explain more about a rule, the student can click on the "Template" button at the bottom of the Rule Browser (Figure 7).

A dialog box comes up, with a partial definition of the rule that has blanks for the student to fill in (see Figure 8). The definition is in terms of the preconditions that need to be verified for the rule to be applied and of the consequences that the application of the rule generates. The Templates design reflects how rules are represented in the Andes knowledge base. As we discussed previously, this design aims to scaffold self-explanations of the forms that are most

frequent in students' spontaneous self-explanations: (i) refine and expand the preconditions of an action and (ii) explicate and infer additional consequences of an action.

Clicking on a blank in a Template brings up a menu of possible fillers (See right template in Figure 8). After completing a Template, the student can select "submit" to get immediate feedback. The SE-Coach retrieves the definition of the corresponding rule from the Andes' knowledge base and uses it to verify the correctness of the student's selections, by matching the rule preconditions and consequences with the fillers that the student chose (see Figure 9).

Each template in the interface reflects the content of a physics rule in the knowledge base, and it is associated to that rule, not to a specific example line. Therefore, adding a new example to the SE-Coach's set does not require defining new Templates, as long as the example solution involves only knowledge already encoded in Andes' rules.

As with the Rule Browser, pilot evaluations were fundamental to improve the usability of Templates. For instance, we discovered that students tended to ignore template fillers that were too verbose, even when they were the obviously correct choices. Also, when the list of possible fillers is too long, students seldom read the items at the bottom, especially if they find a plausible filler earlier in the list. The pilot evaluations also showed that if students are given the choice of accessing a Template or not, they tend not to do it. In the first version of the system (Conati, Larkin, & VanLehn, 1997), once a correct rule was selected the student could click on a *Done* button and quit without filling a template. When this option was available, most students never accessed Templates. When asked why, they said that they did not remember what a Template was, although the experimenter had extensively explained the interface at the beginning of the evaluation session. The simple change of giving only the *Template* choice after rule selection (see Figure 7), increased the percentage of students that filled Templates, despite the fact that students could still close a Template without filling it, by clicking on the *Cancel* button at the bottom of the dialogue box (see Figure 8).

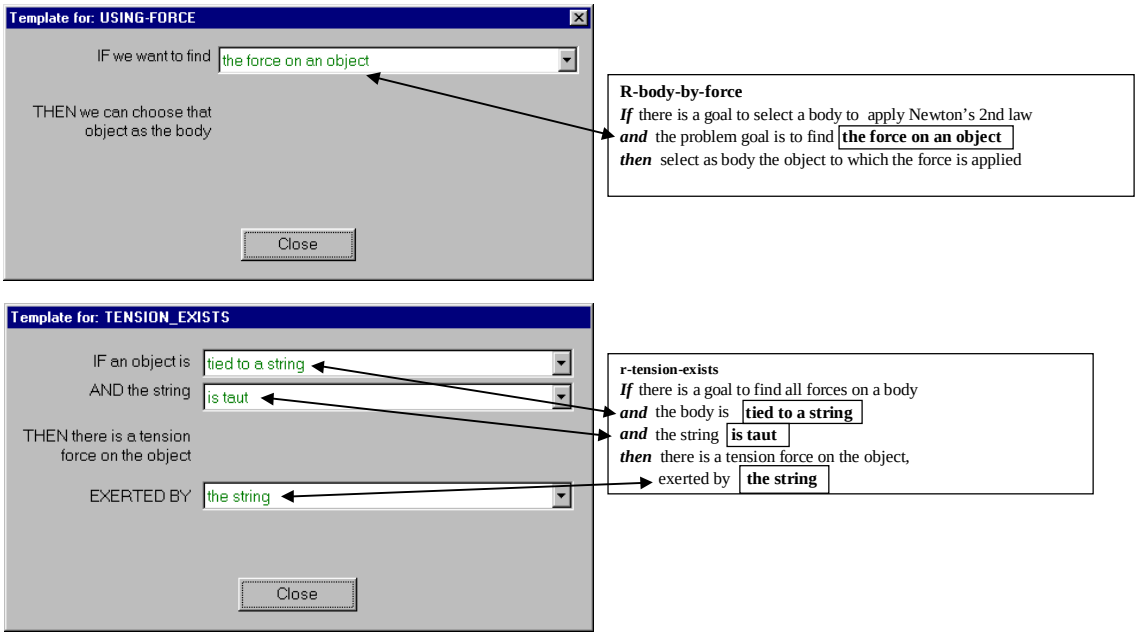

Figure 9: verifying Template correctness by using Andes rules

The Plan Browser

If the student selects the second item in the prompting menus, (e.g., *"The role of this choice in the solution plan is...."* in Figure 6a), then the interface displays a Plan Browser instead of a Rule Browser. The Plan Browser is similar to the Rule Browser, but it displays a hierarchical tree representing the solution plan for a particular example, instead of SE-Coach's physics rules.

For instance, Figure 10 shows the Plan Browser for the 'boy-hanging-from-an-helicopter' examples, which displays the plan to apply Newton's Second Law (Reif, 1995). To explain the role of the uncovered fact in the solution plan, the student navigates through the goal hierarchy and selects a sub-goal that most closely motivates the fact. Pressing a "submit" button causes the SE-Coach to give immediate feedback. To provide this feedback, the SE-Coach retrieves the proposition node corresponding to the uncovered line. If the proposition node encodes a goal, the feedback algorithm works like the one for the Rule Browser. If the proposition node encodes a fact, as the shaded node in Figure 10 does, the feedback algorithm

1. retrieves the goal node that is the most immediate ancestor of that fact (node *G-find-forces* in Figure 10).
2. retrieves the rule node that generates that goal node (node *R-find-forces* in Figure 10).

This rule node, which represents a planning rule, is then used to verify the correctness of the student's selection.

There are no Templates associated with the Plan Browser, because they would simply spell out information on the plan structure already encoded in the Browser hierarchy (e.g., ***If*** *the goal is to apply Newton's law* ***and*** *we have selected a body,* ***then*** *the next subgoal is to describe the properties of this body.*

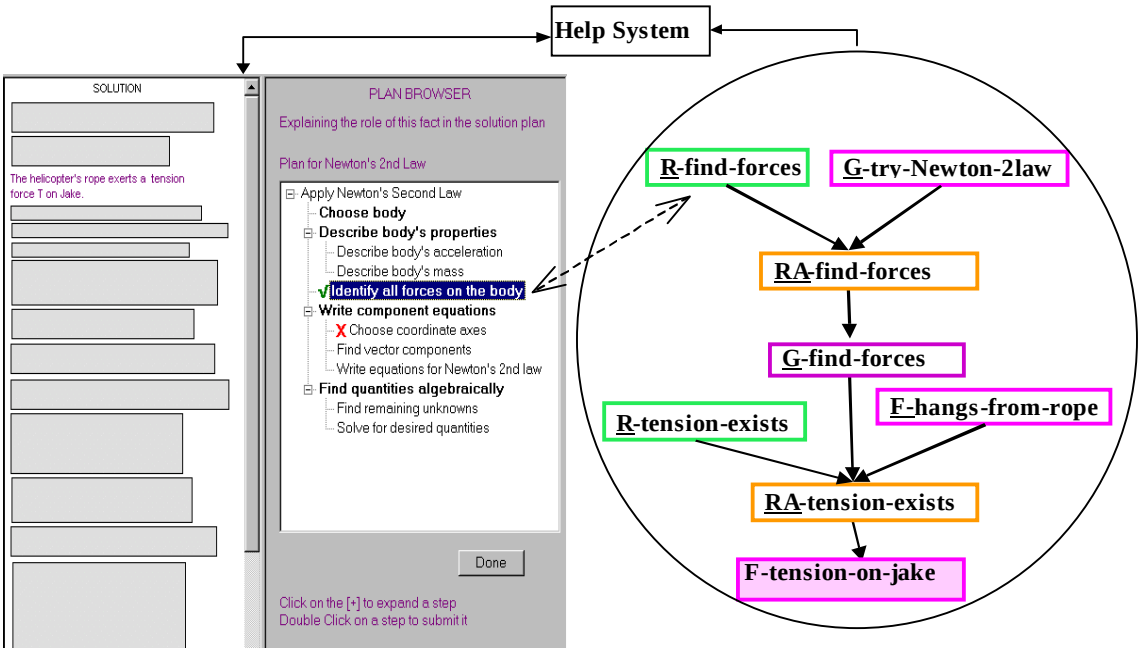

Figure 10: selections in the Plan Browser

SE-COACH'S ADVICE

As we have seen in the previous section, the SE-Coach's interface provides three different levels of scaffolding for self-explanation, implicit in its design. A forth, more explicit level of scaffolding is provided by the SE-Coach's advice.

Initially, self-explanation is voluntary**.** The student is free to decide what self-explanations to initiate with the interface tools, and the SE-Coach limits its intervention to providing feedback. However, the SE-Coach keeps track of the students' progress through the example, including how much time they looked at a solution item and what they chose to self-explain via the interface tools. This information is collected in the SE-Coach's student model, which assesses what parts of the example may benefit from further self-explanation. When the student tries to close the example, the SE-Coach generates tutorial interventions to make the student

self-explain these parts. In this section, we briefly describe how the student model generate its assessment and then we illustrate the tutorial interaction that this assessment supports. More detailed information on the structure and functioning of the student model can be found in (Conati & VanLehn, To appear).

The SE-Coach's student model

Modelling a student during example studying involves a great deal of uncertainty, because the reading and self-explanation actions the model has access to provide only indirect evidence on what the student actually reads and learns. To handle this uncertainty in a principled way, the SE-Coach student model relies on the Bayesian network framework for probabilistic reasoning (Pearl, 1988).

The student model Bayesian network is built automatically when the student opens a new example. Figure 11 summarizes this process. The structure of the Bayesian network derives directly from the structure of the SE-Model. The network parameters derive from probabilities describing the student's physics knowledge and studying style, maintained in the Andes' long term student model. These probabilities provide priors for rule nodes and parameters that automatically define the conditional probabilities in the network. (Conati & VanLehn, To appear).

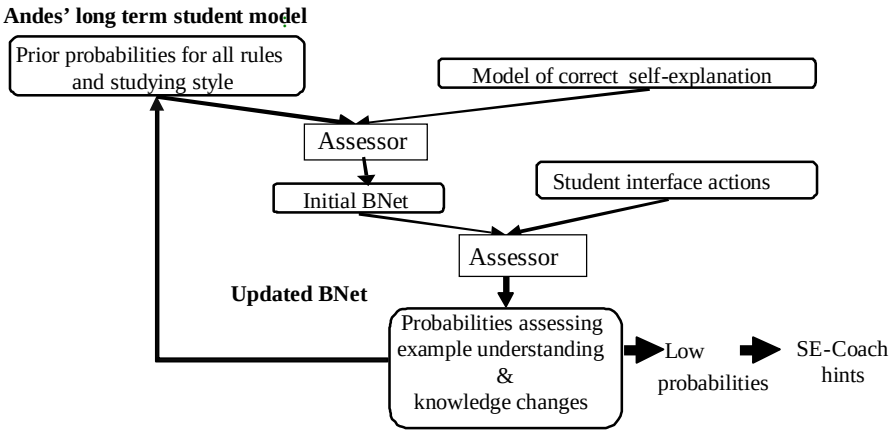

Figure 11: Automatic construction of the student model Bayesian Network.

As students perform reading and self-explanation actions, the initial Bayesian network is updated with nodes and conditional probabilities representing how these actions influence the probability that the student is self-explaining different example components. Nodes representing reading actions directly influence the probability that the student is self-explaining derivations (rule-application nodes) in the SE model. Nodes representing self-explanation actions performed through the interface tools influence the probability that the student knows the corresponding physics and planning rules. Thus, at any time during the student's interaction with the SE-Coach, the probabilities in the Bayesian network assess how the student's knowledge and example understanding change as a consequence of the student's actions. In particular, the probabilities associated with rule-application nodes represent the probability that the student has correctly self-explained the corresponding derivations. Rule-application nodes with probability below a given threshold become the target of the SE-Coach interventions.

A key feature of the SE-Coach student model is that students do not have to use the interface tools to have their self-explanations acknowledged. If a student spends enough time viewing a solution item and if, according to the model, the student has sufficient knowledge to self-explain that item, the model will predict that the student very likely self-explained the item correctly (Conati & VanLehn, To appear). Asking students to always use the interface tools to make their explanations explicit would allow more accurate assessment, but may also burden the students

who are natural self-explainers with unnecessary work, possibly compromising their motivation to use the system.

When a student closes an example, the new probabilities that rule nodes reached during the student interaction with the systems are used to update Andes' long term student model. These probabilities will affect all the subsequent example studying and problem solving interactions of this student with Andes.

The SE-Coach's interventions

As we mentioned at the beginning of this section, while a student studies an example self-explanation is voluntary. However, if a student tries to close the example when the student model indicates that there are still some lines left to self-explain, then the SE-Coach will tell the student:

"*You may learn more by self-explaining further items. These items are indicated by pink covers*".

and colors some of the boxes pink (dark grey in Figure 12) instead of grey. It also attaches to each item a more specific prompt such as "*Self-explain with the Rule Browser*", "*Self-explain with both the Rule and the Plan Browser*" or "*Read more carefully*", depending on what self-explanation the student model predicts to be missing for that item. The more specific prompt appears in place of the simple *self-explain* button when the item is uncovered (see Figure 12). The color of the boxes and the related messages change dynamically as the student performs more reading and self-explanation actions that change the probabilities of the corresponding nodes in the student model. If the student tries to close the example when there are still some pink covers left, the SE-Coach generates a warning such as "*There are still some items that you could self-explain. Are you sure you want to exit?*", but it lets the student quit if the student wants to.

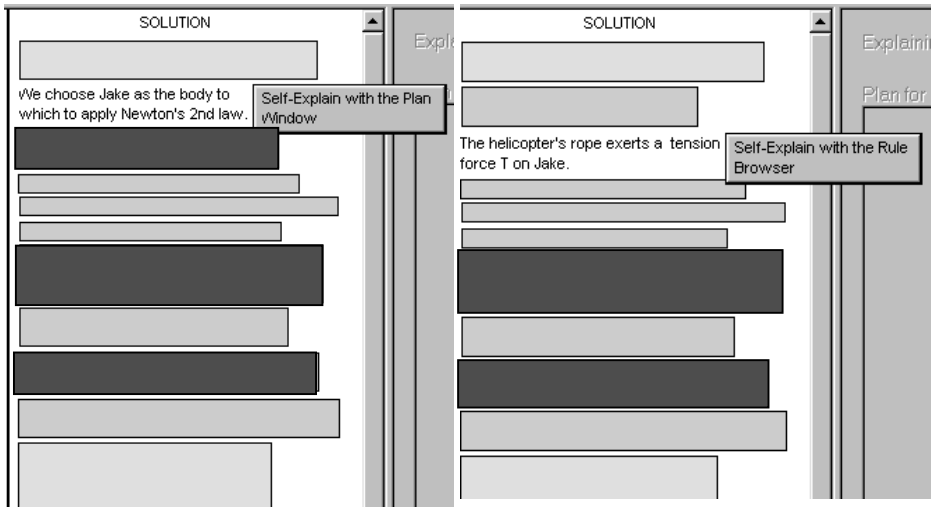

Figure 12: SE-Coach interventions to elicit further self-explanation

As we discussed earlier, one of the challenges of designing the SE-Coach tutorial interventions is that they must motivate to self-explain students that have low propensity to do so. Pilot evaluations were fundamental to find an effective modality of intervention for the SE-Coach. In the original version of the system (Conati et al., 1997), the SE-Coach would point out lines that required self-explanations one at a time, instead of indicating them all at once by changing their color. When the student tried to close the example, the SE-Coach would generate a first, generic warning such as "*There are still some items that you could self-explain. Do you want to try?"* The student could either (a) reject the advice, (b) accept it and go back to study the example without any further indication of what to self-explain, (c) ask for more specific hints. If the student chose the latter, the SE-Coach would say, for instance, "*Why don't you try*

to use the Rule Browser to explain this line?", and it would uncover the line. At this point, the student would go back to the example, and possibly explain the line as indicated, but the only way for the student to get additional suggestions from the Coach would be to try and close the example again.

The rationale behind this design was to stimulate as much spontaneous self-explanation as possible. We thought that directing the student to a particular line in the example could be enough to also trigger explanations on other lines. This did not happen. Either students were natural self-explainers and explained most of the example the first time through, or they strictly followed individual SE-Coach hints but rarely initiated any additional self-explanation. For non-spontaneous self-explainers, the interaction with the Coach would quickly become quite uninspiring. After doing what the Coach had suggested (e.g., finding a rule name in the Rule Browser), they would immediately try to close the example. They would then get another hint ("*there is something else that you could self-explain, do you want me to show you*?"), suggesting further explanation either on the current line via Template/Plan Browser or on a different line. A student would have to repeat this cycle to access each new piece of advice, and most students lost interest and chose to close the example after the first couple of hints.

The current design, based on the coloring of example lines, allows the students to see at once all the parts that they should further self-explain and what interface tools they should use to do it. It also gives students better feedback on the progresses that they are making, because line color and the attached hints change dynamically as students generate more self-explanations.

EMPIRICAL EVALUATION OF THE SE-COACH

After iteratively improving the system design through pilot evaluations, we performed a study to test both the system's usability and its effectiveness for learning.

Experiment design

The study was conducted with 56 college students who were taking introductory physics classes at the University of Pittsburgh (20), Carnegie Mellon University (14), Allegheny County Community College (5) and U.S.Naval Academy (17). The design had two conditions:

- *Self-Explanation (SE)*: 29 students studied examples with the complete SE-Coach.
- *Control*: 27 students studied examples with the masking interface and Plan Browser only. They had no access to the Rule Browser and Templates, nor they received feedback or coaching.

The evaluation consisted of one session about three-hour long, in which students: 1) took a paper and pencil pre-test, consisting of four problems on Newton's Second Law; 2) studied examples on Newton's Second Law with the system; 3) took a paper and pencil post-test with problems equivalent but not identical to the ones in the pre-test; 4) filled out a questionnaire designed to assess the students impressions on the system.

Because SE-Coach does not provide any introductory physics instruction, to evaluate the system adequately we needed subjects who already had the appropriate level of domain knowledge for using it. Students generally benefit more from examples when they are studying a new topic, whereas as the students' knowledge improves, problem solving becomes more effective for learning (Nguyen-Xuan, Bastide, & Nicaud, 1999). Hence, we needed subjects with enough knowledge to understand the topic of the examples, but not so much knowledge to find the examples not worthy of attention. However, because we had to coordinate the study sessions to accommodate students from four different colleges, the best we could do in terms of getting subjects with adequate knowledge was to make sure we run the subjects after their first class on Newton's Second Law and before they took a class test on the topic.

In order to roughly equate time on task, students in the control condition studied 6 examples and students in the experimental condition studied 3 examples. Despite this, there is a statistically significant difference between the average time on task of the experimental group

(52') and the control group (42' 32''). However, we found no significant correlation of time on task with post-test scores.

Usability of the SE-Coach

During the study, we recorded log files of the students' interaction with the SE-Coach. We then analysed the log data to understand how the subjects used the interface self-explanation tools (Rule Browser, Plan Browser and Templates) and how they reacted to the SE-Coach interventions.

Usage of the SE-Coach's self-explanation tools

For each interface tool, we computed the following data summaries, shown in Table 1. *Initiated:* percentage of the explanations that students initiated out of all the explanations that could be generated with that tool for the available examples. *Correct*: percentage of the initiated explanations that were generated correctly. *Attempts before correct:* average number of attempts the students needed to achieve a correct self-explanation. An attempt is the submission of an incorrect self-explanation. *Max # attempt:* average *maximum* number of attempts needed to achieve a correct self-explanation. *Abandoned:* percentage of initiated self-explanations that were abandoned. *Attempts before abandon:* average number of attempts before abandoning a self-explanation. *Time on abandoned:* average amount on time spent on self-explanations that were eventually abandoned.

Table 1: Statistics on SE tools usage

	Rule Browser	**Templates**	**PlanBrowser**
initiated	62%	55.5%	41.6%
correct	87%	97%	85%
attempts before correct	1.27	0.5	1
max # attempts	9.2	2.5	3.8
abandoned	13%	3%	15%
attempts before abandon	4.4	1.9	1.4
time on abandoned	241 sec.	59 sec.	29 sec.

Rule Browser usage. As Table 1 shows, on average students accessed the Rule Browser quite frequently, initiating 62% of the possible Rule Browser explanations. They usually completed the Rule Browser explanations correctly, successfully selecting the correct rule 87% of the times. Also, they usually found the correct rule without floundering too much, requiring on average only 1.27 attempts to find the correct selection. However, for most students at least one rule selection required a large number of attempts (average maximum of 9.2 attempts per student). Rule Browser accesses in which the student failed to find the correct rule required an average of 4.4 attempts. However, these attempts resulted in an average of only 4 minutes that students spent on failed Rule Browser explorations, a minor fraction of the average total time on task (52 minutes).

These data indicate that the Rule Browser was generally successful at stimulating the students to initiate self-explanations and was easy to use. However, there were a few situations in which using the Rule Browser may have caused distraction and frustration, because the student floundered substantially before finding the correct rule, or could not find it at all. Thus, the SE-Coach may benefit from an additional form of help that supports students in searching the Rule Browser. This was, in fact, the suggestion that appeared most frequently in the students' questionnaire. Otherwise, the majority judged the Rule Browser to be very useful and easy to use.

Template usage**.** Students accessed 55.5% of the template material in the SE-Coach. Since Template access is mandatory after a correct selection in the Rule Browser, these data are not indicative of how effectively Templates stimulate self-explanation. More indicative is the fact that students completed almost all (97%) of the presented Templates correctly, although it is not

mandatory to fill a Template after opening it. Students needed on average only 0.5 attempts to fill a Template correctly, with an average maximum of 2.5 attempts. Students spent only 59 seconds trying to fill Templates for which they could not find the correct answer.

These data indicate that our repeated efforts to improve Template usage through pilot evaluations were successful, in terms of both making them easy to use and providing a tool that encourages self-explanation. The results are backed up by the students' questionnaire comments, that judged the Templates to be very easy to use and helpful to better understand the examples.

Plan Browser usage. Students accessed 41.6% of the possible Plan Browser explanations. Students did not have many problems using the Plan Browser. Most of the initiated explanations (85%) resulted in the selection of the correct plan step, and required only 1 attempt, on average, to find the step. Students spent on average only 29 seconds on Plan Browser accesses that did not lead to a correct explanation. However, the fact that an average maximum of 3.8 attempts were needed to find the right selection, indicates that the Plan Browser does produce some degree of floundering. Hence, the Plan Browser could also benefit from a browsing help analogous to the one suggested for the Rule Browser.

Although the above results indicate that the Plan Browser is easy to use, students did not use it quite as much as the Rule Browser and many students wrote in the questionnaire that they did not understand its utility. This outcome is not surprising. As we have already mentioned, goal-related explanations are largely unfamiliar to students. The Plan Browser is designed to complement instruction that initiates students to the notion of solution planning, but our subjects had not received such instruction in the classroom. In the short time available for instructions during the study, the experimenter did not have time to make the students understand the concept and importance of solution planning. This situation likely accounts for the lower usage of this tool.

Response to the SE-Coach interventions to elicit further self-explanations

To verify how students reacted to the SE-Coach explicit prompts to further self-explain (by using the Rule Browser, the Plan Browser or by reading more carefully), we computed from log data how often students followed these prompts. The results are summarized in Table 2. For each type of prompt, the table reports: (i) the maximum number of prompts the SE-Coach could generate for the three examples in the study. These are the prompts the system would generate if there was no student model to guide it; (ii) the number of prompts the SE-Coach actually generated by relying on the student model; (iii) how many of these prompts the students followed.

The numbers in Table 2 show that the current design for the SE-Coach intervention, based on dynamically changing the color of example lines and the attached prompts, works considerably better than the original design described in the sub-section on the SE-Coach intervention. While with the original design students rarely followed more than a couple of the SE-Coach suggestions, with the current design students followed an average of 38.6% of the Rule Browser prompts, 42% of the Plan Browser prompts and 34% of the prompts suggesting to read more carefully. However, the numbers also show that there is still room from improvement, because students ignored more than half of the SE-Coach's suggestions. This could have happened for two reasons.

Table 2: Statistics on SE-Coach interventions

Rule Browser prompts (max. 43)	**22.6**
Followed	**38.6%**
Plan Browser prompts (max. 34)	**22.4**
Followed	**42%**
Reading prompts (max. 43)	**7**
Followed	**34%**

The first is that students did not follow the SE-Coach's suggestions because they had spontaneously self-explained the related example parts. The student model assessment of spontaneous self-explanation strongly depends on estimates of student's initial physics knowledge (Conati & VanLehn, To appear). At the time of the evaluation, we did not have accurate estimates and we assigned to every rule a probability of 0.5. Thus, it may be that students explained more than the model estimated, and were rightly ignoring those prompts asking them for redundant explanations.

The second reason is that students did not follow the SE-Coach's suggestions because they were overestimating their understanding and thus dismissed the SE-Coach's prompts as irrelevant, even when they were well justified. To deal with these students, it might be necessary to make the SE-Coach suggestions mandatory. This option may work better when more accurate priors on the students' physics knowledge are available, for instance from the results of the student's pre-test. However, it would be interesting to try it even with less accurate priors. The benefits of having the students generate more explanations may still be worth the annoyance that redundant prompts may cause to some students.

Effectiveness of the SE-Coach

To evaluate the effectiveness of the SE-Coach for learning, we first compared how the students in the SE condition learned in comparison to the students in the control condition. The pre-test scores of the two conditions held no significant differences, indicating that subjects had been successfully randomised over their physics knowledge.

The gain scores measuring the difference between post-test and pre-test scores were higher for the SE condition, but the difference between gain scores of SE and control conditions was not statistically significant.

We then restricted the analysis to the subgroups of subjects coming from different colleges: Carnegie Mellon University (CMU), Community College of Allegheny County (CCAC), University of Pittsburgh (PITT) and U.S. Naval Academy (USNA). We found that the experimental condition of CMU and CCAC students performed significantly better than the control condition. In contrast, in both the Pitt and USNA subgroups, students in the control condition performed slightly better than students in the SE condition. The commonality of behaviour between CMU and CCAC is quite surprising, because the two schools are supposed to be, respectively, the best and the worst of the four colleges in the study, and this ranking is confirmed by the pre-test scores. However, there is one characteristic that CMU and CCAC have in common and that distinguishes them from Pitt and USNA students. They start the semester at least a week later than Pitt and USNA. Therefore, although all the students participated in the experiment after they had their lectures on Newton's laws and before they took a class test on the topic, Pitt and USNA students were ahead in the course schedule and had likely spent more time on Newton's laws than CMU and CCAC students when they participated in the study. This could have generated differences in the background knowledge and/or in the way the two subgroups used the system, affecting how the students benefited from it.

To test this hypothesis, we continued our analysis by comparing the subgroup consisting of CMU and CCAC students with the subgroup including Pitt and USNA students. Within the CMU-CCAC group, the 12 students in the SE condition (mean gain score = 7.5, st.dev. = 5.2) performed significantly better ($p = 0.021$) than the 13 students in the control condition (mean = 3.2, st.dev.= 3). The 14 Pitt-USNA students in the control condition performed slightly better (mean = 6.7, st.dev. = 4.8) than the 17 in the SE condition (mean = 5.0, st.dev. = 3.7) but the difference is not statistically significant ($p > 0.2$).

Table 3: ANCOVA for post-test scores

Source	F-ratio	Prob
Const	2541	< 0.0001
pretest	54.544	< 0.0001
SAT_math	1.3207	0.2575
SAT_verbal	0.63487	0.4304
subgroup	0.37289	0.545
condition	0.15014	0.7005
subgroup*cond.	7.6089	0.0088

To verify if these results were due to differences in initial physics and background knowledge, we ran an ANCOVA with post-test as dependent variable, subgroup and condition as main factors, and pre-test and SAT scores as covariate. The ANCOVA (see Table 3) still gives a significant interaction between subgroup

and condition ($p < 0.01$), indicating that prior physics and background knowledge do not account for the different behavior of the two subgroups.

To verify if students in the two subgroups used the system differently, we compared their performance and study behavior within each condition. As Figure 14 shows, in the SE condition CMU-CCAC students performed better than Pitt-USNA students, although the difference is not statistically significant ($p > 0.1$). In the Control condition, Pitt-USNA students performed significantly better than CMU-CCAC students ($p < 0.03$). As a matter of fact they performed almost as well as CMU-CCAC students in the SE Condition. These results could be due to two reasons:

- In the SE condition, CMU-CCAC students used the SE-Coach self-explanation tools more extensively and effectively than the Pitt-USNA students did.
- In the Control condition, Pitt-USNA students spontaneously self-explained considerably more that CMU-CCAC students did.

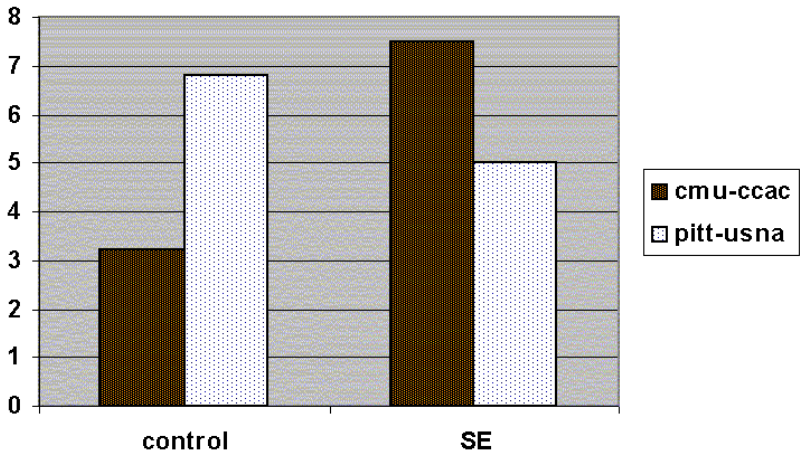

Figure 14: Gain scores for CMU-CCAC and Pitt-USNA students within the two conditions

Studying the behavior of the two subgroups in the SE condition

To verify whether CMU-CCAC students in the SE condition used the SE-Coach self-explanation tools better than the Pitt-USNA students in the same condition, we compared the two subgroups with respect of time on task and the system usage statistics described in the previous subsection. We found a statistically significant difference only in the average number of attempts tried before giving up on a Template explanation (Conati & VanLehn, 2000). This difference suggests that CMU-CCAC students had a higher level of motivation to learn from the SE-Coach, because they started learning Newton's second laws later than Pitt-USNA students and thus perceived studying example on this topic to be more useful, consistently with the findings described in (Nguyen-Xuan et al., 1999).

Browser selections and template filling are recall tasks not as constructive as generating explanations verbally, unless students actively reflect on the result of their actions. We argue that, because CMU-CCAC students were more motivated to study the SE-Coach examples, they reasoned more on what they were doing with the SE-Coach interface. Thus, they learned more, although they did not use the SE tools more frequently or more easily than Pitt-USNA students did. We can corroborate this argument by computing the correlation between: (i) the number of rules in the student model that have reached high probability to be known through correct SE actions; (ii) post-test scores. The correlation is very low ($r < 0.1$) for Pitt-USNA and it is higher ($r = 0.33$) for CMU-CCAC, indicating that correct SE actions reflect knowledge more accurately for CMU-CCAC students than they do for the Pitt-USNA ones.

Studying the behavior of the two subgroups in the Control condition

Verifying whether Pitt-USNA students in the control condition spontaneously self-explained more that CMU-CCAC students is not easy, because students could not communicate their explanations to the control version of the SE-Coach. The only measures that can indicate self-explanation in the log files include: (i) mean and standard deviation of multiple accesses to example lines[1]; (ii) mean and standard deviation of the time spent on each example line; (iii) mean number of accesses and selections in the Plan Browser.

Within the Pitt-USNA control group, we found a marginally significant correlation of post-test scores with mean ($p < 0.09$) and standard deviation ($p < 0.06$) of line accesses (see Table 4).

Table 4

Analysis of Variance For cases selected according to	posttest **Pitt-USNA**	
Source	**F-ratio**	**Prob**
Const	786.68	<0.0001
mean-line-accesses	3.6978	0.0834
pretest	16.488	0.0023
st-dev-line-acecsses	4.6118	0.0573

This correlation is not significant for the CMU-CCAC control group. The hypothesis that Pitt-USNA students self-explained more in the control condition is consistent with the fact that Pitt-USNA students had started studying Newton's Laws earlier and had probably gained more knowledge on the topic. This knowledge was not strong enough to make Pitt-USNA students perform better in the pre-test. However, it was sufficient to enable Pitt-USNA control subjects to generate effective self-explanations under the minimal scaffolding provided by the masking interface. We argue that it is indeed the minimality of the scaffolding that allowed Pitt-CMU control students to bring to bear their knowledge at best. Because of their more advanced learning stage, spontaneous self-explanation triggered by the masking interface likely came quite effortlessly to Pitt-USNA control students. Thus, their tendency to self-explain was not suffocated by the lower level of motivation that prevented Pitt-USNA students in the SE condition to learn effectively from the SE-Coach self-explanation tools.

Summary

The results that we have presented suggest the following conclusions on the effectiveness of the SE-Coach and, in general, on the effectiveness of support for self-explanation during example studying.

- Rich scaffolding for self-explanation, like the one provided by the complete SE-Coach in the experimental condition, can improve students' performance at an early learning stage. At this stage, students are still unfamiliar with the subject matter. Hence, they benefit more from structured help in using domain knowledge to generate effective self-explanations and are more motivated to put substantial effort in exploiting this help at best.
- As students become more proficient in the subject matter, even minimal prompting, like that provided by the masking interface in the control condition, can help improve their self-explanations. At this stage, more elaborate scaffolding can actually be less effective, if it requires students to put too much effort in studying examples, because they may lack the motivation to do so.

Of course, more data are necessary to confirm these conclusions. The data should be gathered in the context of classroom instruction, where it is easier to control at what learning stage students use the system. These data would provide valuable insights on the issue of how much prompting and scaffolding is necessary to elicit self-explanation. Although different studies have shown that both simple prompting and more elaborate scaffolding improve self-explanation (Bielaczyc et al., 1995; Chi et al., 1994), no study has yet addressed the explicit comparison of different kinds of intervention.

[1] A high standard deviation indicates that students were selectively reviewing only some of the example lines to generate their self-explanations.

The study described in this paper did not provide specific results on the effectiveness of the Plan Browser. Limits on the length of the study sessions prevented us from inserting in the pre-test and post-tests items specifically tapping the planning knowledge the Plan Browser is designed to improve. Furthermore, students did not use the Plan Browser as much as they used the other interface tools because they did not understand its function (as most students wrote in the study questionnaire). This was to be expected. The Plan Browser is designed to stimulate reasoning about solution plans from students that already have a basic understanding of the topic. Our subjects most likely did not have such understanding, because they had not received any classroom instruction on solution planning. Thus, many subjects ended up either ignoring the Plan Browser or using it without really understanding the underlying planning knowledge. However, we found a significant correlation (after controlling for pre-test scores) between post-test results and percentage of SE-Coach hints to self-explain with the Plan Browser that the students followed (Conati & VanLehn, To appear). This correlation gives an initial indication that Plan Browser might in fact improve learning, but only a new evaluation with more appropriate subjects and more informative pre/post-test can provide reliable data on this issue.

CONCLUSIONS AND FUTURE WORK

The research presented in this paper represents a step toward exploring innovative ways in which computers can enhance education and learning. Most existing ITS support students during problem solving and teach domain specific skills. We have devised a computational framework that supports learning from examples and that coaches the general learning skill known as self-explanation - generating explanations and justifications to oneself when studying an example. Our framework, known as the SE-Coach, aims to provide the individualized monitoring and guidance to self-explanation that has been proven so beneficial when administered by human tutors. The framework has been implemented and tested within Andes, a tutoring system that helps students learn Newtonian physics through both example studying and problem solving.

We believe that a combination of theoretical foundations and empirical studies is fundamental for the development of effective instructional systems. This is especially true for systems that, like the SE-Coach, focus on a learning process whose underlying mechanisms are still unclear and under investigation. In this paper, we described how the system's design is grounded in Cognitive Science findings about the features that make self-explanation effective and how the design evolved through a careful process of iterative design. In particular, we described two fundamental elements of the system: (a) the SE-Coach interface, that provides specific tools to stimulate and scaffold self-explanation and (b) the SE-Coach's advice, which uses the assessment of a probabilistic student model to elicit self-explanations that can improve the students' example understanding. We conclude by discussing the results of a formal study to test the usability and effectiveness of the system.

Log data from the study indicate that the SE-Coach's interface is easy to use, and that both the interface and the SE-Coach's advice are generally successful in stimulating self-explanation. The analysis of data on students' learning provide interesting initial insights on the educational effectiveness of the SE-Coach and on support for self-explanation. The results suggest that structured scaffolding of self-explanation can be more beneficial at early learning stages, while as students become more proficient in the subject matter, even simpler forms of prompting can successfully trigger self-explanation. Further studies to confirm these findings should be performed in a classroom setting, where it is easier to control when students use the system. These studies would contribute to understanding how much prompting and scaffolding is necessary to effectively elicit self-explanation. This issue is still under investigation in Cognitive Science and is highly relevant to understand what level of sophistication is worth the effort when developing a tutoring system that supports self-explanation.

In principle, the framework underlying the SE-Coach is general enough to support self-explanation in any domain in which problem solving can be formalized as a rule-based cognitive model of the solution process (e.g., math, statistics, geometry, programming, medical diagnosis, troubleshooting). In practice, generating the cognitive model for a new domain can

be very labour intensive, as very labour intensive can be to calibrate the parameters for the probabilistic student model. Do the flexibility and adaptability that these components provide justify their cost? The results that we have presented provide an initial indication that they do, at least in certain learning stages. We plan to seek further evidence by:

- comparing versions of the SE-Coach with and without feedback for correctness, to understand what are the benefits of this feedback for learning.
- comparing versions of the SE-Coach with and without the student model, to evaluate the benefits of targeting the SE-Coach interventions to the students' needs versus making the students generate all the self-explanations relevant to an example.

Another issue that we plan to explore is the role of the masking interface in the self-explanation process. As we discussed in the paper, most of our pilot subjects reported that the masking interface not only did not bother them, but helped them study the examples more carefully. But, of course, these subjective opinions do not provide formal evidence that the masking interface does not interfere with learning. As a matter of fact, we believe that the masking interface does interfere with learning for some students. Therefore, we are exploring the possibility of using eye-tracking to monitor students' attention, given that this technology is becoming increasingly accurate and non-invasive. We plan to perform studies to understand what types of learners benefit more from the interface that uses eye-tracking and what learners benefit more from the additional scaffolding provided by the masking interface, to dynamically adapt the interaction mode to the learner type.

ACKNOWLEDGEMENTS

Our thanks go to the members of the Andes project: A. Gertner, Zendong Niu, Anders Weinstein. We also thank Jill Larkin for her valuable contributions to the design and evaluation of the system and G. Carenini for comments on drafts of this paper. This work was supported by grant number N00014-96-1-0260 from the Office of Naval Research, Cognitive Science Division

REFERENCES

Aleven, V., & Ashley, K. D. (1997). *Teaching case-based argumentation through a model and examples: Empirical evalution of an intelligent learning environment.* Paper presented at the Artificial Intelligence in Education, Kobe, japan.

Aleven, V., Koedinger, K. R., & Cross, K. (1999). *Tutoring answer-explanation fosters learning with understanding.* Paper presented at the AIED '99, 9th World Conference of Artificial Intelligence and Education, Le Mans, France.

Anderson, J. R., et al. (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences, 4*(2), 167-207.

Baker, M. (1999). . Paper presented at the International Conference on Artificial Intelligence in Education, Le Mans, France.

Bielaczyc, K., Pirolli, P., & Brown, A. L. (1995). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem-solving. *Cognition and Instruction, 13*(2), 221-252.

Burrow, R., & Weber, G. (1996). *Example explanation in learning environments.* Paper presented at the Intelligent Tutoring Systems - Proceedings of the Third International Conference, ITS '96.

Catrambone, R. (1995). Aiding subgoal learning: effects on transfer. *Journal of educational psychology, 87,* 5-17.

Chi, M. T. H. (in press). Self-Explaining: The dual process of generating inferences and repairing mental models. *Advances in Instructional Psychology*.

Chi, M. T. H., et al. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 15*, 145-182.

Chi, M. T. H., et al. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18*, 439-477.

Clancey, W., J. (1990). *Knowledge-Based Tutoring: The Guidon Project.*: MIT Press.

Conati, C., Larkin, J., & VanLehn, K. (1997). A computer framework to support self-explanation, *Proceedings of the Eighth World Conference of Artificial Intelligence in Education*

Conati, C., & VanLehn, K. (To appear). *Providing adaptive support to the understanding of instructional material.* Paper presented at the IUI 2001, International Conference on Intelligent User Interfaces, Santa Fe, NM, USA.

Conati, C., & VanLehn, V. (2000). *Further Results from the Evaluation of an Intelligent Computer Tutor to Coach Self-Explanation.* Paper presented at the ITS 2000, 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada.

Ferguson-Hessler, M. G. M., & Jong, T. d. (1990). Studying physics texts: Differences in study processes between good and poor solvers. *Cognition and Instruction, 7*, 41-54.

Gertner, A., Conati, C., & VanLehn, K. (1998). Procedural help in Andes: Generating hints using a Bayesian network student model., *Proceedings of the 15th national Conference on Artificial Intelligence* .

Gott, S. P., Lesgold, A., & Kane, R. S. (1996). Tutoring for transfer of technical competence. In B. G. Wilson (Ed.), *Constructivist Learning Environments* (pp. 33-48). Englewood Cliffs, NJ: Educational Technology Publications.

Koedinger, K. R., et al. (1995). Intelligent tutoring goes to school in the big city. In J. Greer (Ed.), *Proceedings of the 7th World Conference on Artificial Intelligence and Education* (pp. 421-428). Charlottesville, NC: AACE.

Moore, J. D. (1996). Discourse generation for instructional applications: Making computer-based tutors more like humans. *Journal of Artificial Intelligence in Education, 7*(2), 181-124.

Moore, J. D., Lemaire, B., & Rosenblum, J. A. (1996). Discourse generation for instructional applications: Identifying and exploiting relevant prior explanations. *Journal of the Learning Sciences, 5*(1), 49-94.

Nguyen-Xuan, A., Bastide, A., & Nicaud, J.-F. (1999). *Learning to solve polynomial factorization problems: by solving problems and by studying examples of problem solving, with an intelligent learning environment.* Paper presented at the *AIED '99, 9th World Conference of Artificial Intelligence and Education*, Le Mans, France.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan-Kaufmann.

Pirolli, P., & Recker, M. (1991). Knowledge construction and transfer using an intelligent tutoring system: The role of examples, self-explantion, practice and relfection. .

Pirolli, P., & Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction, 12*(3), 235-275.

Ploetzner, R., & Fehse, E. (1998). Learning from explanations: Extending one's own knowledge during collaborative problem solving by attempting to

understand explanations received from others. *International Journal of Artificial Intelligence in Education, 9*.

Reif, F. (1995). *Understanding basic mechanics*: Wiley & Sons.

Renkl, A. (1997). Learning from worked-examples: A study on individual differences. *Cognitive Science, 21*(1), 1-30.

Renkl, A., et al. (1998). Learning from worked-out examples: the effects of example variability and elicited self-explanation. *Contemporary educational psychology, 23*, 90-108.

Ryan, R. (1996). *Self-explanation and adaptation.* , University of Pittsburgh, Pittsburgh.

VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *ITS96: Proceeding of the Third International conference on Intelligent Tutoring Systems.* . New York: Springer-Verlag.

VanLehn, K., Jones, R. M., & Chi, M. T. H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*(1), 1-59.

Vivet, M. (1987). *Meta-knowledge and Explanations in Intelligent Tutoring Systems.* Paper presented at the Second European Conference for Research on Learning and Instruction, Tubingen, Germany.

Webb, N. M. (1989). Peer interaction and learning in small groups. *International Journal of Educational Research, 13*, 21-40.

Weber, G., & Specht, M. (1997). *User modeling and adaptive navigation support in WWW-based tutoring systems.* Paper presented at the Proceedings of User Modeling '97.

Woltz, U., McKeown, R., & Kaiser, G. (1990). Automated Tutoring in Interactive Environments: a task-centered approcah. *Machine-Mediated Learning, 3*(1), 53-79.