The Design and Experimental Evaluation of a Scaffolded Software Environment to Improve Engineering Students' Disciplinary Problem-Solving Skills

Christopher Hundhausen*, Pawan Agarwal*, Richard Zollars†, and Adam Carter*

Human-centered Environments for Learning and Programming (HELP) Lab

*School of Electrical Engineering and Computer Science

†Voiland School of Chemical Engineering and Bioengineering

Washington State University

P.O. Box 642752

Pullman, WA 99164-2752

hundhaus@wsu.edu

Abstract

Background

Introductory gateway engineering courses are notorious for their high attrition rates. Deficiencies in students' *problem-solving processes* may contribute to their failure in these courses. In an empirical study of student problem solving, we observed that students struggle because of misconceptions regarding the basic syntax and semantics of disciplinary diagrams and corresponding mathematical equations.

Purpose (Hypothesis)

We hypothesize that a scaffolded software environment that provides dynamically-generated feedback on the syntactic and semantic correctness of students' evolving disciplinary diagrams and mathematical equations can improve engineering students' problem-solving abilities.

Design/Method

We iteratively developed ChemProV, a software environment to assist chemical engineering students in solving material balance problems. To evaluate ChemProV's effectiveness, we performed two between-subjects experimental studies. The first study compared a preliminary version of the ChemProV to pen-and-paper. The second study compared a redesigned version of ChemProV with dynamic feedback to the same version of ChemProV without dynamic feedback.

Results

While it did not uncover any significant differences, the first study provided insight into how to improve ChemProV's dynamic feedback mechanism. The second study found that the "feedback" version of ChemProV promoted a statistically-significant advantage in problem-

solving accuracy, significantly more time-on-task, and a transfer-of-training to an unscaffolded problem-solving situation.

Conclusions

A scaffolded software environment like ChemProV can serve as a valuable aid in helping students learn engineering problem-solving skills. Its software design approach can be used as a model for designing educationally-effective software environments for other engineering disciplines.

Keywords

Problem solving, software scaffolding, visualization

1. Introduction

The demand for engineering graduates is on the rise, yet engineering degree programs continue to struggle to retain the students who enter their programs. Indeed, introductory gateway courses typically lose 35 percent of their students (Gainen, 1995), and fewer than 50 percent of students who enter engineering degree programs go on to graduate (Borrego, Padilla, Zhang, Ohland, & T. J. Anderson, 2005). These data suggest students' success or failure in an engineering program is related to their ability to pass introductory gateway courses. Such courses typically introduce students to an engineering approach to solving disciplinary problems—an approach they will need to use throughout their careers as engineers (see, e.g., Felder, 1986). Why, then, do so many students fail to complete introductory gateway courses?

1.1 Possible Reasons for Failure

Student failure in gateway engineering courses has been linked to many factors, ranging from deficiencies in teacher training and pedagogical approaches (see, e.g., Custer & Daugherty, 2009), to the lack of an adequate learning community (see, e.g., Besana & Dettori, 2004), to individual student differences (see, e.g., Bergin, Reilly, & Traynor, 2005), to mismatches in cognitive learning styles (see, e.g., Felder & Spurlin, 2005). Another possibility, based on past research into differences in novice and expert problem-solving (Chi, Glaser, & Rees, 1982; Gick, 1986; see, e.g., Bransford, Brown, & Cocking, 1999), is that deficiencies in students' *problem-solving processes* contribute to their failure in gateway courses. In order to explore how students go about solving disciplinary problems, and why they fail, we conducted an exploratory empirical study in which we recorded four pairs of introductory chemical engineering students as they solved material balance problems (Zollars, Hundhausen, & Stefik, 2007). This process involves three key steps: (a) creating a process flow diagram that models the chemical process

described in the material balance problem; (b) creating a corresponding system of mathematical equations; and (c) solving the equations for the unknowns.

In reviewing the video recordings, we found one area in which all pairs had difficulties: *translating the problem statement into a process flow diagram and set of mathematical equations*. None of the groups was able to put together a correct process flow diagram. Without a correct process flow diagram, the derivation of the appropriate material balance equations was impossible. Common errors included omitting critical components, symbolizing material streams as processing units, and adding components beyond those that were described in the problem statement. We viewed students' inability to translate a problem statement into a proper process flow diagram and corresponding set of material balance equations as the fundamental problem to be addressed.

While our exploratory study focused on material balance problems in chemical engineering, engineering contains many of these kinds of problems, which we call *engineering translation problems*. One typically solves such problems by first translating them into disciplinary diagrams, and then using the diagrams as a basis for deriving sets of mathematical equations. For example, electrical engineers use circuit diagrams to solve problems involving electricity, and civil and mechanical engineers use free body diagrams to solve problems involving statics. Empirical research into physics problem-solving (see, e.g., Larkin, McDermott, D. Simon, & H. Simon, 1980) and mathematics problem-solving (see, e.g., Superfine, Canty, & Marshall, 2009) suggests that the ability to translate between problem representations is a hallmark of expertise. Thus, we have reason to believe that students in other engineering disciplines suffer from translation problems analogous to the ones we observed in our study of chemical engineering students.

1.2 A Possible Solution

Grounded in the learning theory of Vygotsky (1978), *software scaffolding* aims to modify problem-solving tasks such that they are more manageable for learners (see, e.g., C. Quintana et al., 2004). A common approach is to provide learners with additional support and guidance in the form of coaching, task structuring, feedback, and hints (see, e.g., Guzdial, 1994). In many science, engineering, and math education domains, the approach has proved successful in helping students learn problem-solving skills (see, e.g., Roschelle and Stroup; Metcalf, Krajcik, and Soloway; VanLehn et al. 2002).

Given the difficulties we observed in our exploratory study, we hypothesized that a software scaffolding approach could help engineering students who are first learning to solve engineering translation problems. In particular, we suspected that dynamically-generated feedback with respect to the syntactic and semantic correctness of students' evolving diagrams and equations could gently guide students toward correct solutions, and ultimately help them to develop essential problem-solving skills that they would need not only in gateway courses, but throughout their careers as engineers.

1.3 Research Questions

Our interest in the development of a computer-based tool to scaffold the process of solving engineering translation problems raises at least three key research questions.

RQ1: How can we design a computer-based tool that scaffolds the process of solving engineering translation problems and ultimately enables students to develop the skills necessary to solve the problems on their own?

RQ2: Will such a scaffolded tool promote better problem-solving accuracy than tools without scaffolding?

RQ3: Will such a tool promote a transfer-of-training that allows students ultimately to retain their problem-solving skills without the scaffolding provided by the tool?

1.4 Article Preview and Outline

To address these questions, we present the design evolution and experimental evaluation of a software tool designed to scaffold the process of solving engineering translation problems within the domain of chemical engineering. The software tool, ChemProV (Chemical Process Visualizer), scaffolds the process of solving material balance problems in three key ways: (a) by providing a toolbox of components that ease the task of constructing syntactically-correct process flow diagrams, (b) by enabling students to drag-and-drop variables in their process flow diagrams directly into equations; and (c) by providing dynamically updated feedback messages that alert students of errors in their process flow diagrams and equations, and provide hints on how to fix them. While it did not uncover any significant performance differences, an experimental comparison of a preliminary version of ChemProV and pen-and-paper provided insight into how to improve ChemProV's dynamic feedback mechanism. After redesigning ChemProV based on those insights, we ran a more focused follow-up experimental evaluation that compared a redesigned version of ChemProV with feedback against the same version of ChemProV without feedback. This study found that ChemProV promoted a statistically-significant advantage in problem-solving accuracy, significantly more time-on-task, and a robust transfer-of-training when compared to the no-feedback version of ChemProV.

Thus, the main contribution of the work presented here is twofold: (1) a novel scaffolded software environment for chemical engineering education whose design can be leveraged to improve students' skills in solving similar engineering translation problems in other engineering disciplines; and (2) experimental evidence of the software environment's educational

effectiveness, including evidence that it promotes a transfer-of-training to an unscaffolded situation.

The remainder of this article is organized as follows. Section 2 provides a brief introduction to material balance problems, and reviews related work. Section 3 presents our preliminary design of ChemProV. Section 4 presents an experimental evaluation of a preliminary version of ChemProV, which led to the redesign of ChemProV's dynamic feedback mechanism. Section 5 presents the redesigned version of ChemProV, together with an experimental evaluation of its dynamic feedback. Finally, Section 6 summarizes our work, considers its implications for engineering education, and outlines directions for future research.

2. Background and Related Work

A focal point of chemical engineering is the modeling of chemical processes. Chemical engineers model these processes using *process flow diagrams*, which depict chemical streams flowing through process units that change the chemical composition of the streams. For example, Figure 1 depicts a simple process flow diagram in which two streams enter a mixer unit. The mixer unit combines the two streams into a single stream, which is fed to a separator unit. The separator unit separates its input into two output streams.

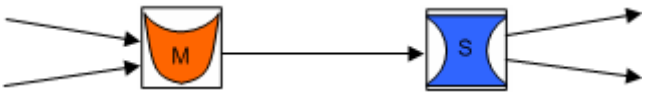

Figure 1. A simple process flow diagram

In analyzing a process flow diagram like the one above, chemical engineers are interested in formulating *mass balances* to describe the chemical process and solve for unknown quantities. In the simplest case (the one addressed by the ChemProV software), the chemical process is both

steady-state (i.e., it does not change over time) and *continuous* (i.e., the input/output streams are continuously fed/removed). As a result, when formulating mass balances, one can apply the fundamental *law of conservation of mass*: the sum of the masses of the input streams equals the sum of the masses of the output streams. In addition, the masses (moles) of individual molecular species (in the case of nonreactive systems) or atomic species (in the case of reactive systems) can also be balanced.

ChemProV supports the construction of process flow diagrams that model continuous, steady-state processes, and the formulation of systems of mass balance equations based on those process flow diagrams. Several commercial chemical process simulation environments also support these tasks, including Aspen, HYSYS, CHEMCAD, Chempute, GAMS, and PRO/II.(*PRO/II*, n.d.) These commercial software environments differ from ChemProV in three key respects:

1. *Designed for professionals, not learners.* While they are frequently introduced in chemical engineering curricula, especially at the senior level (Dahm, Hesketh, & Savelski, 2002), commercial chemical process simulation tools are specifically designed to increase the productivity of chemical engineering professionals who need to construct accurate simulations of chemical processes. In contrast, ChemProV is designed specifically for chemical engineering students first learning about material balances.
2. *Equations solved in background.* Commercial chemical process simulation environments, because of their focus on easing the simulation task, formulate and solve material balances automatically in the background. As Dahm et al. (2002; see also Clough 2000) point out, this sets up a situation in which "it is possible for students to successfully construct and use models without really understanding [them]" (*p*. 192). In contrast, because ChemProV aims

to help chemical engineering students learn how to solve material balance problems, ChemProV requires students to construct their own process flow diagrams and material balances, providing educational scaffolds to assist students in this process.

3. *Powerful feature set*. Commercial chemical process simulation environments are powerful tools, offering an array of sophisticated features, including a large palette of processing operations (e.g., distillation, absorption), a large palette of thermodynamic packages, and a variety of techniques for dealing with recycle streams. These features would overwhelm an introductory chemical engineering student, and are beyond the scope of an introductory chemical engineering course. In contrast, ChemProV supports the simplest of chemical processes (continuous and steady state), and is tailored specifically for students first grappling with the basic concepts of material balance problems.

While we are unaware of existing educational software environments specifically designed to support the process of solving material balance problems, there is a legacy of research into educational software environments that assist students in learning and problem-solving in science and engineering domains. We find it useful to classify the environments in this body of work into two broad categories, according to the ontological status they assume in the learning process. At one end of the spectrum is a family of *intelligent tutoring systems* that aim to assist learners by acting as *expert tutors* (Anderson, Boyle, and Reiser 1985; Anderson et al. 1995). To do this, they maintain a cognitive model of the learner during the learning process, providing dynamic feedback and guidance based on that model. In empirical evaluations, the best intelligent tutoring systems have proved to be highly effective in promoting problem-solving skills in several domains, including math (e.g., Koedinger et al. 1995) and computer

programming (e.g., Anderson, Conrad, and Corbett 1989). To our knowledge, no intelligent tutoring system has been developed specifically to promote engineering problem-solving skills.

ChemProV is similar to intelligent tutoring systems in that it provides students with dynamic feedback messages to guide learners toward correct problem-solving actions. However, unlike cognitive tutors, ChemProV does not maintain a cognitive model of the learner, and has no knowledge of the exact problem being solved. Rather, its feedback system is based purely on the syntactic and semantic rules of disciplinary representations—process flow diagrams and corresponding systems of material balance equations. In this sense, it more closely resembles the "minimalist AI" approach advocated by Suthers (1999).

At the other end of the spectrum from cognitive tutors is a family of learning environments that assume the role of a *mediational resource* (Roschelle, 1996) rather than that that of an expert teacher or coach. Rooted in social constructivist learning theory (see, e.g., Lave and Wenger 1991) and the notion of "cognitive apprenticeship" (Collins, J.S. Brown, & Newman, 1989), these environments take the view that learners can best learn *collaboratively, socially, and actively* by undergoing a process of inquiry, constructing representations, and using those representations to discuss and reason about the target concepts and skills (see Jonassen & Land, 2000). Thus, the focus of this family of tools is on providing appropriate tools to foster educationally-beneficial inquiry, discourse, and reflection, rather than on providing the kind of guidance that might be given by a coach or teacher.

A key example in this line of work is a group of general *concept-mapping tools* that enable collaborating learners to graphically represent and discuss their evolving conceptions of a problem inquiry in any domain of science or engineering (see, e.g., Suthers, Toth, and Weiner 1997; Luchini, Quintana, and Soloway 2003). In addition, several environments have been

constructed for specific science and engineering disciplines, including physics (Roschelle, 1996), chemistry (Michalchik et al., 2008), biology (Douglas, Peterson, & Udovic, 1998), and computer science (Hundhausen and Brown 2008). While some of the tools in this body of work have been constructed specifically to mediate face-to-face collaborative discourse (see, e.g, Roschelle 1996; Douglas, Peterson, and Udovic 1998; Hundhausen and Brown 2008), there has been increasing interest in building tools for synchronous (see, e.g., Suthers, Hundhausen, and Girardeau 2003) and asynchronous (see, e.g., Suthers and Xu 2002) online discourse. All of these tools share ChemProV's goal of assisting students in learning new concepts and problem-solving skills. However, the emphasis of this family of tools is on supporting collaborative student learning through inquiry, communication and discussion, rather than on guiding individual learners toward the construction of correct solutions.

The overarching goal of the ChemProV software is to improve learners' problem-solving skills, so that they can ultimately solve material balance problems without the aid of the software. A large body of related research shares this goal of helping learners develop independent problem-solving skills. Within this body, one influential line of research has carefully considered how to sequence instruction so as to best promote the acquisition of problem-solving skills (for an overview, see Ritter, Nerb, Lehtinen, & O'Shea, 2007). For example, Renkl and Atkinson (2003) introduce the strategy of *fading worked examples*, in which learners begin by studying fully worked-out problem examples, and subsequently solve isomorphic problems in which worked-out solution steps are gradually removed. In several experimental studies, this strategy has proved to be significantly more effective than the classic approach of presenting learners with a worked-out example followed by a problem to solve (Renkl & Atkinson, 2007). More recently, the approach has been successfully applied to the

design of problem-solving exercises involving computer-based intelligent tutoring systems (see, e.g., Schwonke et al., 2007). In the research studies presented here, students were introduced to material balance problems through a short series of lectures, and they completed a brief software tutorial that walked them through how to use the ChemProV software. While our studies did not consider alternative ways of ordering examples and problems so as to facilitate learning, we are optimistic that such approaches could enhance learners' ability to benefit from the ChemProV software.

Another body of related research has explored specific learning environment features that can promote successful problem solving. For example, one line of work considers the design of on-demand help and how students use it (see, e.g., Aleven, Stahl, Schworm, Fischer, & Wallace, 2003). Another line of research more closely aligned to work presented here focuses on identifying the kinds of feedback that can best promote successful problem solving. In a recent comprehensive review of this research, Shute (2008) extracts key guidelines for the design and use of feedback to enhance learning and problem-solving. ChemProV's dynamic feedback mechanism adheres to many of these guidelines—most notably, (a) "focus feedback on...specific features of [the learner's] work in relation to the task, with suggestions on how to improve" (p. 177), (b) elaborate on problems rather than simply verifying correctness, and (c) present "specific and clear" feedback messages in "manageable units" (p. 177).

One specific form of feedback identified in the review of Shute (2008) as particularly appropriate for novice learners is *scaffolding*. Based on Vygotsky's (1978) theoretical notion of the "zone of proximal development," scaffolding aims to empower learners to engage in more advanced problem solving than they otherwise could. In order to make the process of solving material balance problems more manageable for chemical engineering students, ChemProV

makes use of *software scaffolding* (see, e.g., Guzdial 1994). Quintana et al. (2004) present an integrative review of the large body of software scaffolding approaches, and derive a framework of guidelines for designing educationally-effective software scaffolding. The design approach taken by ChemProV employs at least four strategies suggested by the guidelines in this framework: (a) “embed expert guidance,” (p. 347), (b) “make disciplinary strategies explicit in [the software interface]” (p. 351) (c) “enable learners to inspect multiple views of the same object or data,” (p. 354) and (d) “give learners ‘malleable’ representations” that can be directly manipulated (p. 355).

3. Preliminary Design of ChemProV

Though an iterative, learner-centered design process (see, e.g., Soloway et al. 1996) , we have developed ChemProV (Chemical Process Visualizer) to assist chemical engineering students in solving material balance problems. Our design process began with a preliminary study of student problem-solving with pen-and-paper (Zollars, Hundhausen, and Stefik 2007). Aside from illuminating learners' problem solving processes, the study uncovered several common errors that learners commit when solving material balance problems, including (a) omitting critical components of process flow diagrams, (b) symbolizing material streams as processing units, (c) adding components beyond those that were described in the problem statement; (d) incorrectly expressing the contents of material streams; and (e) incorrectly formulating material balance equations.

3.1 Overview of Environment

These observations led to the development of a software environment that assists students in solving material balance problems by scaffolding the tasks of constructing process flow diagrams and corresponding material balance equations. Figure 2 presents a snapshot of the

ChemProV environment, with key components of the environment annotated in blue. To place a process unit (mixer or separator) into the Diagram Area, users simply select the corresponding tool in the Toolbox and click within the Diagram Area. To create chemical streams that flow through process units, users simply select the Chemical Stream tool, and then drag out a stream, possibly connecting it to a process unit. To specify the chemical composition of a stream, users populate the Stream Tag Table associated with the stream. In the first row of the Stream Tag Table, users must give the stream an overall label, quantity and units (unknowns are specified with a '?'); subsequent rows can be used to specify the quantity, units, and type of compound of each chemical component of the stream.

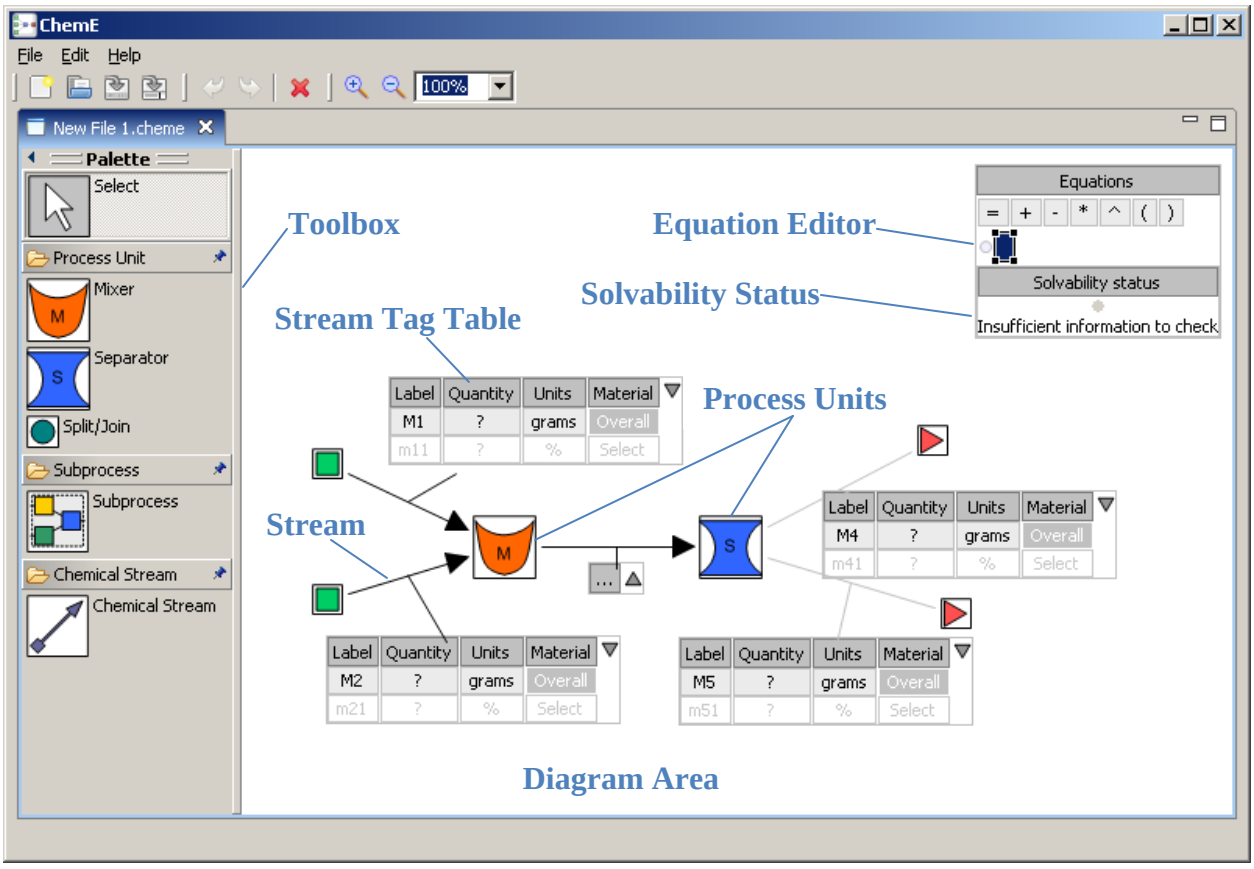

Figure 2. Annotated screenshot of ChemProV

Users specify material balance equations in the Equation Editor. They can do this in one of two ways: by dragging and dropping stream component labels and mathematical operators (located at the top of the Equation Editor itself) into equations in the equation editor, or by directly typing in equations from the keyboard. As equations are entered, ChemProV continuously checks the equations to see whether they are independent and solvable for a solution, and the Solvability Status is updated accordingly.

3.2 Dynamic Feedback Messages

In analyzing the errors that students made in the preliminary study, we found that most errors stemmed from students' failure to heed the fundamental syntactic and semantic rules for constructing valid process flow diagrams and material balance equations. To help students identify and correct these errors, ChemProV continously monitors students' solutions as they are being constructed, and generates dynamic feedback messages whenever the rules are violated. The feedback messages appear initially as warning icons placed at the location of a rules violation—either within a process flow diagram or next to an equation. When students move the mouse over the warning icons, the feedback messages appear.

The syntactic and semantic rules on which ChemProV's dynamic feedback messages are based fall into three broad categories:

1. *Diagram consistency.* Based on the conservation of mass principle, the first category of rules pertain to chemical compound streams within process flow diagrams. For each process unit in a process flow diagram, the types, quantities, and units of the input compounds must match the types, quantities, and units of the output compounds. For

example, Figure 3 depicts a feedback message that is generated when the units and quantities of the incoming and outgoing compounds of a process unit do not match.

2. *Equation correctness*. The second category of feedback messages are based on the basic algebraic rules for material balance equations. First, each equation specified in the Equation Editor must have consistent units (e.g., all grams or all gm/min.). Second, in an equation that expresses an overall balance across a process unit, only the terms of the overall streams (as opposed to the individual components of the streams) may appear in the equation (see, e.g., Figure 4). Third, in an equation that expresses the balance of an individual component within a stream, only the individual component, possibly expressed as a percentage of the overall flow rate, may be included in the equation.

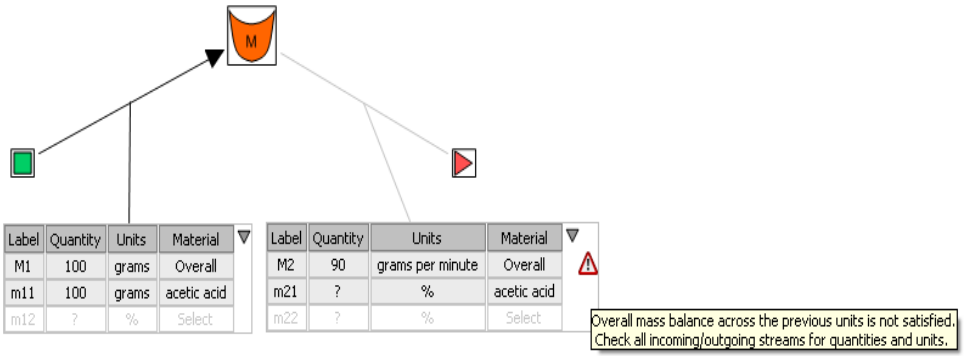

Figure 3. Feedback message generated when units and quantities of incoming and outgoing streams do not match

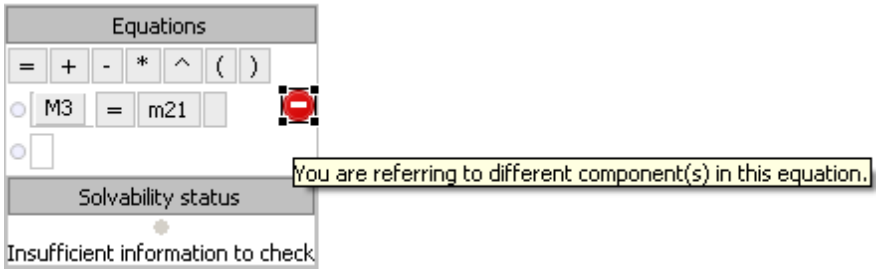

Figure 4. Example of feedback message when invalid terms are used to refer to an overall balance across a process unit

3. *Equation solvability*. A final category of feedback messages, which appear in the Solvability Status area of ChemProV, relate to whether or not the system of equations as a whole is solvable. The first rule in this category is that there must be a sufficient number of equations to solve for all unknown values present in the process flow diagram. In other words, each unknown element in the process flow diagram must be represented in at least one equation (see Figure 5). The second rule in this category states that the set of equations must be *independent*; that is, values must exist for all variables that enable all equations to be simultaneously satisfied. If both of these rules are satisfied, then ChemProV presents a message indicating that the set of equations is solvable (see Figure 6).

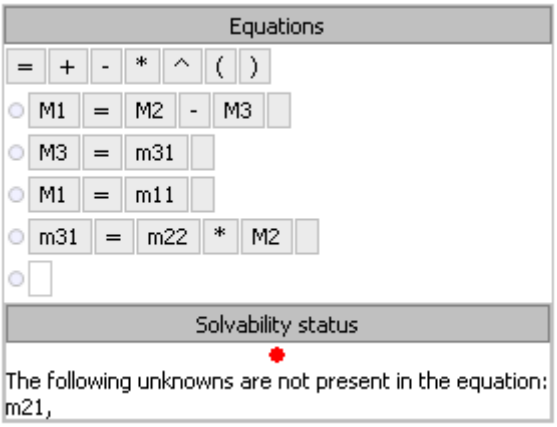

Figure 5. Solvability status message that appears when not all unknowns that appear in the process flow diagram are present in the system of equations

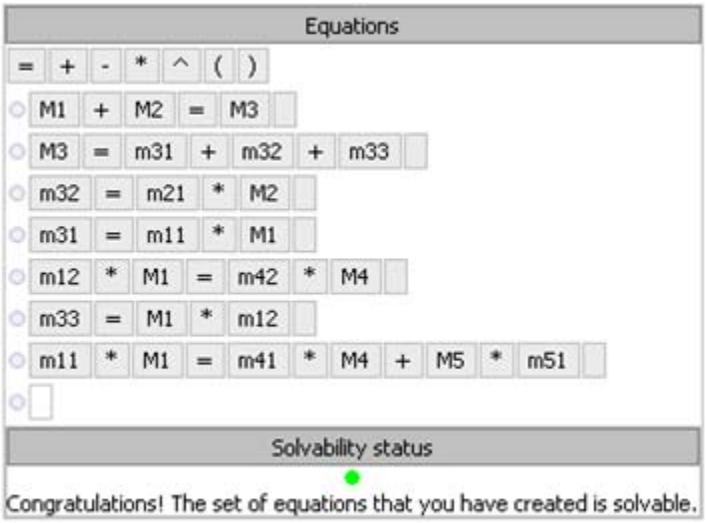

Figure 6. Solvability status message that appears when set of equations is solvable

3.3 Summary

In summary, ChemProV aims to help students develop problem-solving skills by scaffolding the problem-solving process in three key ways:

- by providing a constrained toolbox of process flow diagram components (mixers, separators, chemical streams), and constraining how they can be composed;
- by allowing equations to be built by dragging-and-dropping stream component labels from process flow diagrams into equations; and
- by presenting dynamic feedback messages that both alert students of errors in their diagrams and equations, and provide hints on how to fix them.

As illustrated in Table 1, these forms of scaffolding align well with five strategies for educationally-effective scaffolding suggested by the framework of Quintana et al. (2004).

Table 1. Ways in which scaffolding strategies described by Quintana et al. (2004) are manifested in ChemProV. Strategy labels are those used by Quintana et al. in their article, and page numbers refer to pages in the article on which the strategy is first presented.

Scaffolding Strategy	**Manifestation in ChemProV**
1c: Embed expert guidance to help learners use and apply science content (*p*. 347)	ChemProV presents dynamic feedback messages that alert students of syntactic and semantic errors in their process flow diagrams and equations; feedback messages disappear when the corresponding errors are corrected
2a: Make disciplinary strategies explicit in learners' interactions with the tool (*p*. 351)	The toolbox of process flow diagram components encourages creation of valid process flow diagrams; chemical stream tables provide template for specifying valid chemical streams; variables in process flow diagrams can be dragged-and-dropped into equations to encourage creation of valid equations
3b: Enable learners to inspect multiple views of same object or data (*p*. 355)	Both graphical (process flow diagram) and mathematical (equations) representations can be inspected side-by-side, allowing students to see correspondences between the two
3c: Give learners "malleable" representations that can be directly manipulated (*p*. 355)	Process flow diagrams can be created and edited by direct manipulation; equations can be created and edited either by direct keyboard entry or by dragging-and-dropping variables from process flow diagrams
4a: Restrict a complex task by setting useful boundaries for learners (*p*. 360)	ChemProV models only steady-state, continuous chemical processes consisting of a constrained set of process elements (mixers and separators)

4. Experimental Evaluation of Preliminary Version of ChemProV

In order to evaluate the educational effectiveness of the version of ChemProV described in the previous section, we experimentally compared it to pen-and-paper, the medium traditionally used to solve material balance problems. The goal of ChemProV's software scaffolding is both to enable learners to solve material balance problems more easily, and to enable learners ultimately to maintain the same level of problem-solving performance without the scaffolding. Accordingly, we formulated the following two hypotheses for our experimental comparison:

H1: Students will be able to create more accurate solutions to material balance problems with ChemProV than with pen-and-paper.

H2: The problem-solving proficiency that students attain with ChemProV will transfer to pen-and-paper.

To test these hypotheses, we conducted a matched-samples between-subjects experimental study with two conditions: ChemProV First and Pen-and-Paper First. Participants in each condition solved two material balance problems of roughly equivalent difficulty. For the first problem, participants used the tool corresponding to their condition (ChemProV or pen-and-paper). For the second problem, participants switched to the other tool (ChemProV or pen-and paper).

In a study of human performance such as this one, a within-subjects design would generally be advantageous, because it gracefully handles individual differences, which are known to be significant, especially in studies of novice performance. However, in this study, we anticipated there would also be a strong *learning effect* from problem one to problem two, regardless of the tool used first. Thus, we opted for a between-subjects design, with the two conditions matched as well as possible. While such a design sacrificed experimental power and introduced the possibility that individual differences would mask actual differences in the effectiveness of each tool, it enabled us to measure both the *initial accuracy* with each tool, untainted by a the learning effect we anticipated, as well as the transfer-of-training effect between tools, as predicted by H2.

4.1 Participants

We recruited 21 students (15 male, 6 female; mean age 19.7) out of the Spring, 2009 offering of ChE 110, "Introduction to Chemical Engineering," at Washington State University. Participants were recruited in the tenth week of the semester, after they had received a week of preliminary instruction on material balance problems. Participants received course credit for their participation.

4.2 Materials and Tasks

All participants completed two problem-solving tasks: one using the ChemProV described in the previous section, and another using pen-and-paper. When they used ChemProV, participants worked on a Pentium IV computer running the Windows XP operating system. When they used pen-and-paper, they were given a black pen and sheets of paper to work out their solutions (see Figure 7).

Prior to working with each of the tools (ChemProV or pen-and-paper), participants completed tutorials that introduced them to that tool. The tutorials were designed to be informationally-equivalent, in order to ensure that neither tutorial afforded a problem-solving advantage.

All participants completed two material balance problems designed to be of equivalent difficulty. Both problems involved two processing units. One problem, Problem A, had total 15 variables, of which the values of seven variables were unknown. The other problem, Problem B, had total 16 variables, of which the values of eight variables were unknown. For both problems,

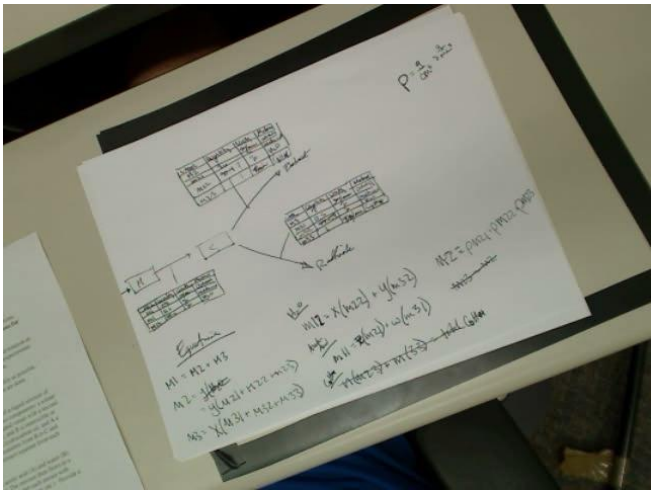

Figure 7. Experimental set-up for pen-and-paper condition

participants were instructed to create a process flow diagram and set of equations to solve for all the unknown variables in the system; however, they did not actually have to solve the equations to obtain a solution.

We used Morae® Recorder to make lossless recordings of participants' screens as they worked on tasks. These recordings allowed us to recreate participants' work if needed, and to measure their time on task.

4.3 Procedure

All 21 students enrolled in ChE 110 participated in the study as a required part of the course. In order to assign participants to the two conditions, we first rank-ordered participants based on their course grades at the time of the study (week 10 of the semester). We then assigned participants to the two conditions in a round-robin fashion, such that the two conditions were optimally matched with respect to their grades. This resulted in 10 participants being assigned to the ChemProV condition, and 11 participants being assigned to the Pen-and-Paper condition. To guard against order effects, we counterbalanced problem order within each condition, such that half the participants in each condition completed Problem A first, and the other half completed Problem B first.

Participants completed the study one-at-a-time within a quiet laboratory setting. Study sessions lasted two hours apiece. At the beginning of the study sessions, participants signed an informed consent form, and were given a brief overview of, and general set of instructions for, the study. To ensure that all the participants received the same information, we prepared a script that was read verbatim to all the participants. Next, participants completed the two problems, using a different tool (ChemProV or pen-and-paper) for each problem. Prior to working on each problem, participants were given 15 minutes to complete a tutorial on the tool they would be

using to complete the problem. Participants were given up to 40 minutes to solve each problem. They were asked to solve each problem as quickly as possible, without sacrificing accuracy.

4.4 Measuring the Dependent Variables

To measure time on task, we reviewed the video recordings of all the participants, noting the time at which students began and stopped their work. To measure the overall accuracy of participants' solutions, we devised a solution scoring scheme that considered the accuracy of three distinct components of students' solutions. Each solution component was weighted to reflect its relative importance in the final solution:

- *Process flow diagram layout accuracy* (weighted 20% of overall solution score). Participants' process flow diagrams were checked to ensure that they had all of the required components (mixer, separator, and streams), and were composed properly. One point was awarded for each correct process unit and stream; one point was subtracted for each incorrect unit or stream.
- *Process flow diagram stream accuracy* (weighted 30% of solution score). The compositions of the streams in participants' process flow diagrams were assessed for their accuracy and completeness. Each correct entry was awarded one point, missing entries received zero, and incorrect entries were penalized a point.
- *Material balance equation accuracy* (weighted 50% of solution score). Participants' material balance equations were checked for accuracy and completeness. An equation was *completely correct* (1 point) if all elements of the equation—the units, material and operators—were correct. An equation was *conceptually correct* (¾ point) if it included all of the correct compounds, but had incorrect units. An equation was *partially correct* (¼ point) if one or more components were omitted from the balance. An equation was

incorrect (0 points) if it did not fall into any of the above categories. Equations that were correct but non-independent were penalized ½ point, while incorrect equations beyond the required equations in the model solution were penalized one point.

To verify the reliability of our grading scheme, we had two chemical engineering instructors independently grade a random sample consisting of 20 percent of the solutions produced in the study (eight solutions to Problem A, and eight solutions to Problem B). The two graders achieved an overall agreement of 95 percent. Based on this result, we concluded that our scoring system was sufficiently reliable, and we had a single instructor grade the remainder of the solutions.

4.5 Results

Table 2 and Table 3 present participants' mean percent correct and time-on-task by condition and problem-solving task. Figure 8 and Figure 9 present box plots of the same data. These box plots reflect a large amount of variance in the data.

Table 2. Mean solution accuracy (% correct) by condition and task number (std. dev. in parentheses)

Condition	*n*	Task 1	Task 2
ChemProV First	10	57.83 (22.62)	69.96 (29.79)
Pen-and-Paper First	11	60.10 (23.05)	61.06 (22.89)

Table 3. Mean time-on-task (in minutes) by condition and task number (std. dev. in parentheses)

Condition	*n*	Task 1	Task 2
ChemProV First	10	32.84 (5.27)	26.58 (7.65)
Pen-and-Paper First	11	28.16 (5.77)	27.35 (10.00)

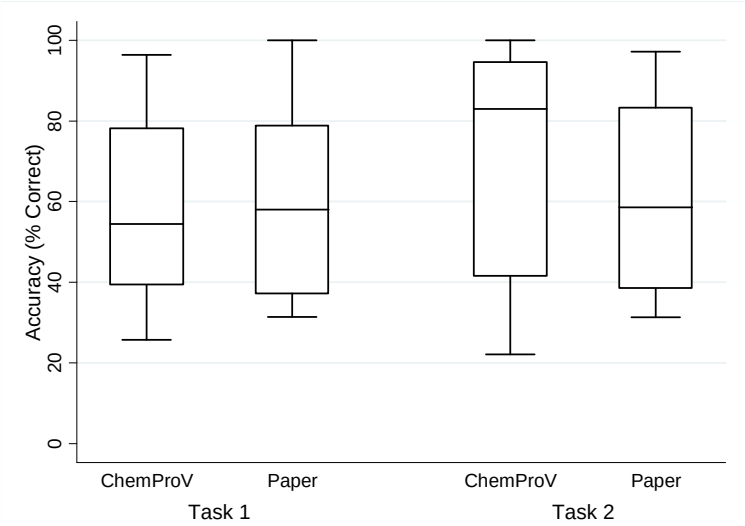

Figure 8. Box plots of solution accuracy by condition and task

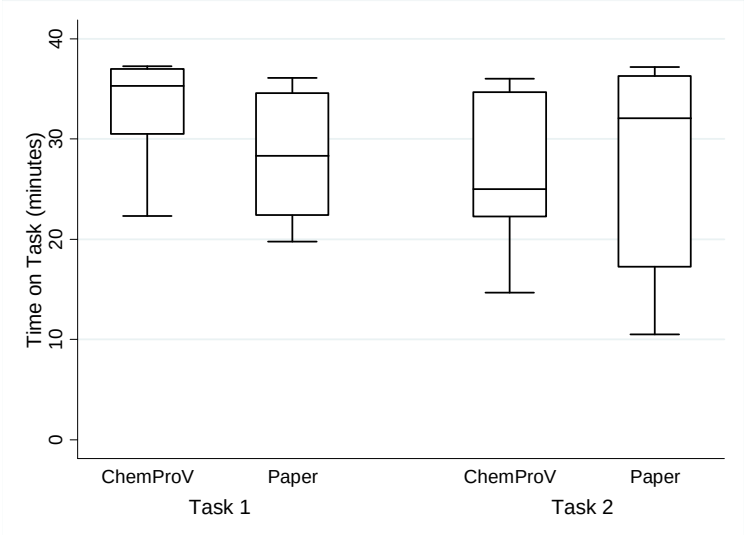

Figure 9. Box plots of time-on-task by condition and task

Before running statistical tests, we used Shapiro-Wilk tests, coupled with normal probability plots, to confirm that our data were normally distributed. In addition, to guard against the possibility of Type I error, we set the threshold *p*-value at 0.025—0.05 divided by two, the number of tests we were running on each dependent variable. To test H1, we ran a one-tailed (directional) independent-samples *t*-test on our accuracy data. According to that test, there existed no statistically-significant differences between the two conditions with respect to either

Task 1 accuracy, $t(19) = -0.23$, $p = 0.59$, or Task 2 accuracy, $t(19) = 0.77$, $p = 0.45$. While we did not predict a difference with respect to time-on-task, we nonetheless ran two-tailed (non-directional) independent-samples *t*-test on our time-on-task data. That test did not yield any statistically-significant differences between the two conditions with respect to either Task 1, $t(19) = 1.93$, $p = 0.07$, or Task 2, $t(19) = -0.20$, $p = 0.84$.

To test H2, we first performed a paired-sample (within-subjects) *t*-test to compare the Task 1 accuracy of the ChemProV First condition against their Task 2 accuracy. The two-tailed *t*-test found no significant differences between these participants' Task 1 and 2 accuracy, $t(9) = -1.56$, $p = 0.15$, thus providing preliminary evidence that a transfer-of-training took place.

Next, on the assumption that *complete transfer* takes place when one tool can be replaced with the other with no loss of accuracy, we used the following formula, adapted from Singley and Anderson (1989), as a means of measuring *degree of transfer*:

$$\textit{degree of transfer} = \frac{\text{accuracy}_{\textit{pen-and-paper}}}{\text{accuracy}_{\textit{ChemProV}}}$$

Complete transfer would take place with a degree of transfer value of 1.0, with values between 0.0 and 0.99 indicating incomplete transfer, and values above 1.0 indicating "super transfer." Applying this formula to our accuracy data, we obtain a mean degree of transfer value of 1.21 ($SD = 0.51$) for the ChemProV First condition. This indicates that the participants in the ChemProV First condition experienced a more-than-complete transfer-of-training.

4.6 Discussion

Our results provide limited empirical support for our two hypotheses. Counter to the prediction made by H1, participants who used the ChemProV interface first did not significantly outperform participants who used the pen-and-paper interface first in either task. Interestingly, in Task 2, in which the ChemProV First participants switched to pen-and-paper and the Pen-and-Paper First participants switched to ChemProV, the ChemProV First participants improved upon their Task 1 performance by 12 percentage points, and even outperformed the Pen-and-Paper First participants by eight percentage points. This result, reflected in the high degree-of-transfer value we found (1.27), provides evidence for the prediction made by H2. However, it is notable that the Pen-and-Paper participants also experienced a small positive transfer-of-training effect (1.03). This suggests that some of the transfer-of-training effect we observed is likely attributable to a practice effect.

How can these inconclusive results be explained? Why did participants perform so poorly with ChemProV not only in absolute terms (M = 59.50%, SD = 22.25%), but also in comparison to their performance with pen-and-paper (M = 64.80%, SD = 26.28%)? Below, we consider three alternative explanations for our results.

One possible explanation is that the ChemProV interface differentially impacted participants based on their chemical engineering abilities. Indeed, ChemProV is designed especially to help lower-performing students, who are most vulnerable to dropping out of chemical engineering courses. In a follow-up analysis, we partitioned participants into two groups, High (n = 12) and Low (n = 9), based on whether their final grades in the introductory chemical engineering course lay above or below the class average. We found that the Low group performed 4.77 percentage points (SD = 19.57%) better with ChemProV than with pen-and-

paper, whereas the High group performed 12.81 percentage points worse (*SD* = 24.66%). Based on a one-tailed (directional) *t*-test, this difference was statistically significant, $t(1) = -1.76$, $p = 0.047$. See Agarwal (2009) for further details on this result.

A second possible explanation is that the design of ChemProV's software scaffolding is somehow deficient, failing to provide sufficient guidance to help learners succeed in solving chemical balance problems. The most prominent form of scaffolding within ChemProV is its system of dynamic feedback. One might ask whether the feedback messages actually impacted participants' behavior. To explore this issue, we analyzed the event logs generated by the ChemProV software. These event logs kept track of the feedback messages that (a) appeared, (b) were read by participants (as indicated by their placing a mouse over a warning icon to reveal the pop-up error message, which was not otherwise visible), and (c) were fixed by participants (as indicated by the error icon disappearing because the rule associated with the error icon was no longer violated).

We found that ChemProV generated an average of 65.4 feedback messages (*SD* = 49.2) per problem-solving session. Participants explicitly read just 15.75% (*SD* =27.19%) of those messages. They ultimately fixed 93.90% (*SD* = 4.84%) of the messages that were generated. That participants could fix such a high percentage of the messages generated was encouraging. However, we wondered how they could address such a high percentage of feedback messages without explicitly *reading* them. Further exploration of the video footage yielded two notable observations. First, the mere presence of an error icon was enough, in some cases, to motivate participants to diagnose and fix errors without actually reading the details of the associated error message. Second, we observed that participants were able, in some cases, to fix errors without

even noticing the error. This was especially true in cases in which the resolution to the error was clearly evident to the participant (e.g., a units mismatch, a syntactic error in an equation).

A third possible explanation was that participants were not given explicit instruction on how to use the feedback messages in ChemProV to their advantage. Indeed, while the ChemProV tutorial provided step-by-step instructions on how to build process flow diagrams and construct equations, it did not explicitly describe the dynamic feedback messages or how they might aid in the problem-solving process. As a result, it could have been the case that some participants did not even notice the warning icons that appeared. Even if they did figure out what the message icons represented, it could have been the case that some participants did not know they could place the mouse over a message icon to reveal an associated error message. Based on our log data, five of the 21 participants failed to place their mouse over a message icon even once; another four participants did so exactly once. If participants were not aware of the feedback system or its benefits, it would stand to reason that the feedback would have had little benefit.

In sum, this experimental study furnished some evidence that skills gained with ChemProV might transfer to pen-and-paper; however, it failed to provide empirical evidence that ChemProV significantly improves students' problem-solving accuracy. Our investigation into the reasons for these mixed results provide us with insights into how ChemProV might be designed so as to provide better support for student problem-solving. We now turn to follow-up research that sought to improve the effectiveness of ChemProV.

5. Experimental Evaluation of Redesigned Version of ChemProV

Motivated by the results of the experimental study just reported—especially the finding that participants explicitly read only 15.75 percent of the feedback messages—we decided to

revisit the design of ChemProV's dynamic feedback mechanism. We observed that, in order to read feedback messages in ChemProV, participants had to explicitly place the mouse over a warning icon, and that this was something that participants did not often do. This observation aligns well with the findings of Quintana et al. (2002), who observed that scaffolds that had to be triggered by user interaction were not frequently used. Following the recommendation of Quintana et al. (2002) to provide *continuously visible* scaffolds, we decided to redesign ChemProV's feedback mechanism to include a dedicated window that continuously displays feedback messages, thus making the messages more prominent. In this section, we describe this new design, and we present a more focused experimental study to evaluate its educational effectiveness.

5.1 Redesigned ChemProV

Figure 10 presents a screenshot that illustrates the new feedback display mechanism in the redesigned ChemProV software. Spanning the bottom of the screen, a new Feedback Messages Window displays a numbered list of the feedback messages, with the most recently-generated message appearing at the top. Each message also appears as a numbered icon adjacent to the element of the process flow diagram or equation to which it pertains. In order to emphasize the correspondence between the feedback messages that appear in the Feedback Messages Window and those that appear adjacent to process flow diagram elements or equations, ChemProV supports a two-way highlighting mechanism. When a user clicks on a message in the Feedback Messages Window, (a) the message is highlighted in the Feedback Messages Window, (b) the corresponding message icon adjacent to a process flow diagram element or equation is highlighted, and (c) a pop-up message displaying the text of the message appears adjacent to the corresponding message icon. This highlighting mechanism works in the other direction too:

When a user clicks on a message icon adjacent to a process flow diagram element or equation, (a) the icon is highlighted; (b) a pop-up message displaying the text of the message appears adjacent to the message icon, and (c) the corresponding message in the Feedback Messages Window is highlighted. These forms of highlighting go away as soon as the user clicks elsewhere.

In the redesigned ChemProV, the text of the feedback messages also received an overhaul, based on our observation that the messages were not always written in clear, consistent language that would be readily understandable by a learner. In particular, we rewrote the messages such that they each included the same two components: (a) a concise description of the problem, followed by (b) a brief suggestion as to how to fix the problem.

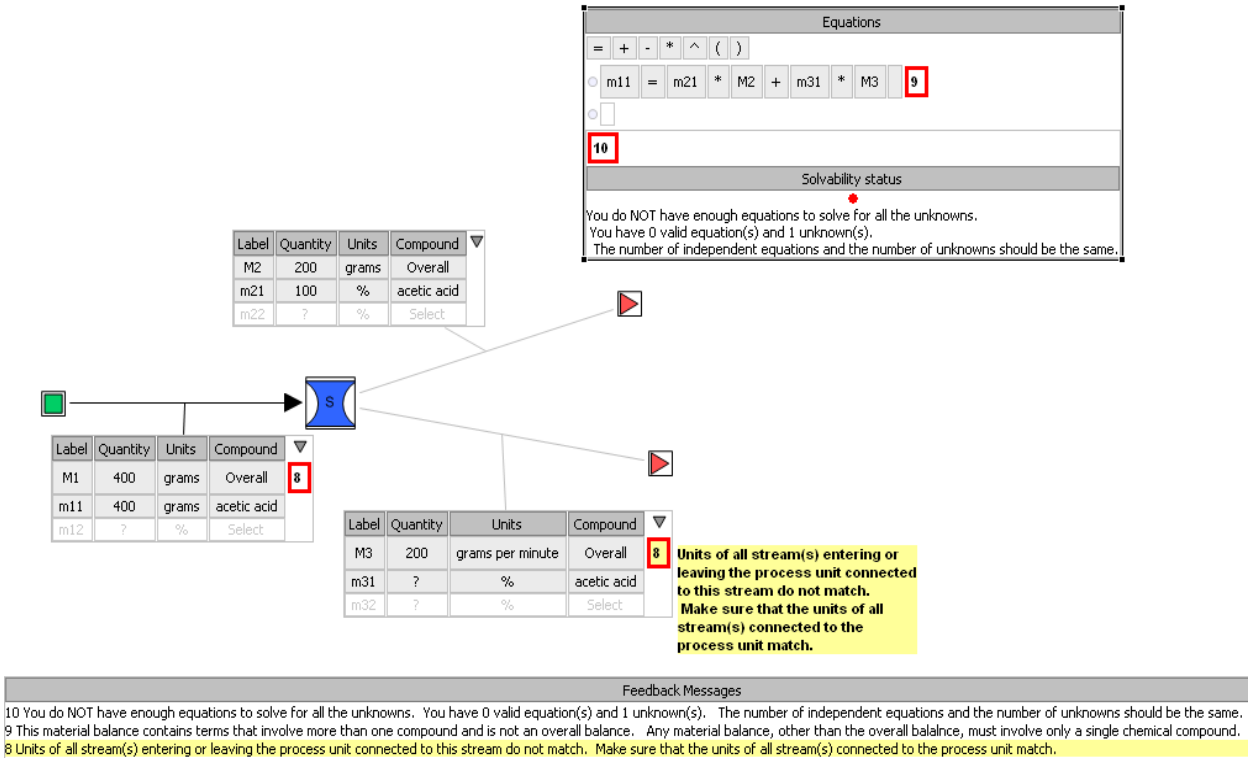

Figure 10. Snapshot of the new dynamic feedback mechanism in the redesigned version of ChemProV

In order to verify the effectiveness of the new display mechanism for feedback messages and the wording of the feedback messages themselves, we ran a small pilot study in which we presented three introductory chemical engineering students with a series of ChemProV screens that presented the feedback messages in an actual context of use. For each screen, we asked participants to describe, in their own words, the problem reported by the feedback message, along with how they would fix the problem, and how they might improve the feedback message. This study enabled us to fine-tune the feedback messages and their presentation prior to our experimental evaluation of the redesigned ChemProV, to which we now turn.

5.2 Experimental Evaluation

To evaluate the effectiveness of the redesigned ChemProV, we conducted an experimental study whose design and protocol were similar to that of the study reported in the previous section, with a two notable exceptions. First, rather than comparing the redesigned ChemProV to pen-and-paper, we opted for a more focused study that could isolate the effects of the redesigned ChemProV's dynamic feedback mechanism, which we hypothesized to be the software scaffolding feature with the greatest impact. To isolate the effects of the dynamic feedback mechanism, we compared a version of ChemProV with the (redesigned) dynamic feedback turned on against the same version of ChemProV with the dynamic feedback turned off.

Second, based on our observation that the ChemProV tutorial in our prior experiment failed to provide any training on how to use the dynamic feedback messages, we added two additional steps to the "feedback" version of the tutorial. These steps had participants (a) intentionally introduce an error into their process flow diagram, (b) observe the feedback message, and (c) fix the error to eliminate the feedback message. Embedded in these additional

steps was the following instruction, which was designed to encourage participants to pay attention to the feedback messages: "As you work through a problem with this version of ChemProV, your goal is to eliminate all feedback messages."

As in our prior experiment, we formulated two hypotheses for this experiment:

H1: Students will be able to create more accurate solutions to material balance problems with a version of ChemProV with dynamic feedback messages than with the same version of ChemProV without dynamic feedback messages

H2: The problem-solving proficiency that students attain with the "feedback" version of ChemProV will transfer to the "no feedback" version of ChemProV.

Similar to the previous experiment, to test these hypotheses, we employed a matched-samples between-subjects design with two treatments defined by the version of ChemProV used in the first task: Feedback First and No Feedback First. Participants in each condition solved two material balance problems of roughly equivalent difficulty; the problems were nearly identical to those used in the first study. For the first problem, participants used the tool corresponding to their condition (the "feedback" or the "no feedback" version of ChemProV). For the second problem, participants switched to the other tool.

As in the previous experiment, our study defined two dependent variables: *solution accuracy* and *time-on-task*, and we additionally collected data on participants' subjective impressions of each tool through an exit questionnaire whose analysis is beyond the scope of this article.

5.2.1 Participants

We recruited 32 participants (21 male, 11 female, mean age 20.5 years) out of ChE 201 "Chemical Process Principles and Calculations," the introductory material/energy balance course at Washington State University designed for students with some prior experience in chemical engineering. ChE 201 differs from ChE 110 (out of which participants were recruited for the first

study) in that ChE 110 covers a variety of topics (the nature of chemical engineering versus other engineering disciplines, ethics, professionalism) whereas ChE 201 focuses on building problem solving skills and material/energy balances; it is a prerequisite for all subsequent ChE courses.[1] Participants took part in the study sometime during the fifth through eighth weeks of the semester, after they had received a week of preliminary instruction on material balance problems. Participants received course extra credit for participating in the study. Since comparisons between the two cohorts were not being made this difference in the level of maturity was not an important factor in our analysis.

5.2.2 Materials, Tasks, and Procedure

The materials, tasks and procedure were identical to those of the previous study, with the following two exceptions. First, as described above, we modified the tutorial for the "feedback" version of ChemProV so that it provided brief instruction on how to utilize the dynamic feedback messages. Second, we slightly modified the wording of one of the problems, because we observed in the first study that some participants had more difficulty with it than with the other problem. In addition, in an attempt to make the two problems more equivalent, we required participants to specify the composition of all streams in their solutions.

5.2.3 Measuring the Dependent Variables

As before, we measured time-on-task by reviewing the video recordings. To measure solution accuracy, we used a solution scoring scheme similar to that used in the previous study, with one notable difference: the scoring scheme was modified to accommodate the small changes in the requirements of the problem solutions described above.

[1]While it is true that the ChE 201 students who took part in this study may have been slightly more advanced than the CheE students who took part in the first study, we believe that any differences in their level of maturity were negligible, especially since students in both studies were provided with an equivalent introduction to material balance problems.

5.2.4 Results

Table 4 and Table 5 present participants' mean percent correct and time-on-task by condition and problem-solving task. **Figure** Figure 11 and Figure 12 present box plots of the same data. As in the previous study, these box plots reflect a large amount of variance in the data, as is typical of novice performance.

Table 4. Mean solution accuracy (% correct) by condition and task number (std. dev. in parentheses)

Condition	*n*	Task 1	Task 2
Feedback First	16	86.70 (11.25)	87.88 (9.70)
No Feedback First	16	69.76 (23.27)	76.63 (18.60)

Table 5. Mean time-on-task (in minutes) by condition and task number (std. dev. in parentheses)

Condition	*n*	Task 1	Task 2
Feedback First	16	31.70 (6.88)	19.67 (5.78)
No Feedback First	16	25.20 (8.47)	25.24 (7.35)

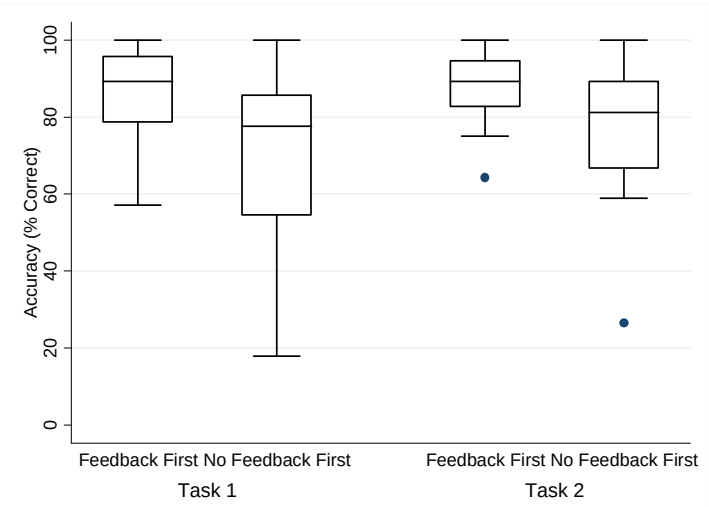

Figure 11. Box plots of solution accuracy by condition and task

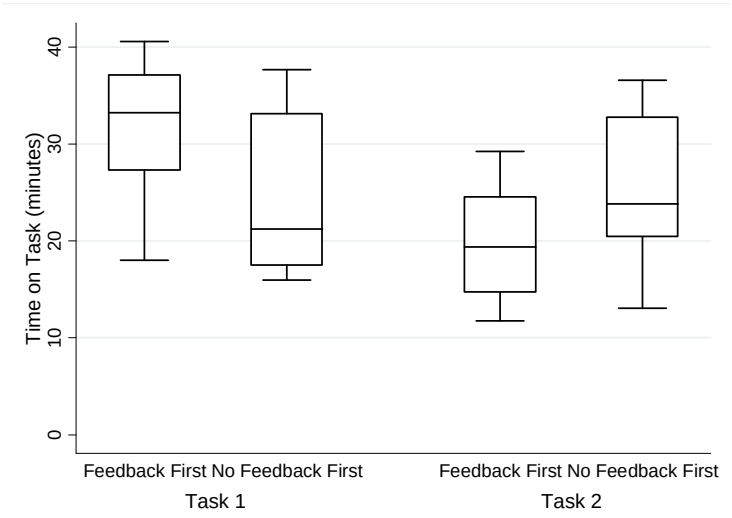

Figure 12. Box plots of time-on-task by condition and task

Before running statistical tests, we used Shapiro-Wilk tests, coupled with normal probability plots, to confirm that our data were normally distributed. We tested H1 with one-tailed (directional) independent-samples *t*-tests on the Task 1 and Task 2 accuracy data. To correct for the possibility of Type I error, we set the threshold *p*-value at 0.05 divided by 2 (0.025). In addition, because the variances between the two conditions were found not to be equal, we used Welch's correction. According to the *t*-tests, the Feedback First condition achieved significantly higher accuracy in both Task 1, $t(22.53) = 2.62$, $p = 0.008$, and Task 2, $t(23.63) = 2.15$, $p = 0.021$. The effect sizes (r^2 values) were modest: 0.19 for Task 1, and 0.16 for Task 2.

Since we did not make a specific prediction with respect to time-on-task, we ran two-tailed (non-directional) independent-samples *t*-tests on these data, once again setting the threshold *p*-value to 0.025 to correct for Type I error. The *t*-tests revealed that the No Feedback First completed Task 1 significantly faster than the Feedback First condition, $t(30) = 2.38$, $p = 0.024$, whereas the Feedback First condition completed Task 2 significantly faster than the No

Feedback First condition, t(30) = -2.39, $p = 0.024$. Again, the effect sizes for both of these differences were found to be modest: 0.16 for Task 1, and 0.15 for Task 2.

To test H2, we first performed a paired-sample (within-subjects) *t*-test to determine whether the Feedback First condition's accuracy differed significantly from Task 1 to Task 2. The two-tailed *t*-test found no significant difference, $t(15) = -0.53$, $p = 0.70$, thus providing preliminary evidence that a transfer-of-training took place.

Next, as in our first study, we computed the mean degree of transfer from Task 1 to Task 2 for the Feedback First condition, obtaining a value of 1.09—an indication that a robust (more than complete) transfer-of-training took place between the "feedback" and "no feedback" versions of ChemProV.

5.2.5 Discussion

In contrast to the results of our first study, the results of this study furnish empirical support for both our hypotheses. Consistent with H1, participants who used the "feedback" version of ChemProV in Task 1 achieved significantly higher task accuracy than participants who used the "no feedback" version of ChemProV. Moreover, in accordance with H2, the problem-solving skills of students in the Feedback First condition transferred to the task they performed without feedback, with a mean degree of transfer of 1.09. This result is bolstered by the finding that the Feedback First condition maintained a significant accuracy advantage in Task 2, despite the fact that (a) they switched to the "no feedback" version of ChemProV, (b) their counterparts in the No Feedback First condition switched to the "feedback" version of ChemProV in Task 2 and also improved on their Task 1 performance by nearly seven percentage points.

Although we failed to predict it a priori, we also detected a significant time-on-task difference in both tasks: Participants who used the "feedback" version of ChemProV completed tasks significantly slower than participants who used the "no feedback" version of ChemProV. In retrospect, we might have predicted this difference. When presented with feedback messages, learners need to read them, think about them within the context of their evolving solutions, and ultimately respond to them by modifying their solutions in some way. It stands to reason that engaging in these kinds *reflective* activities—widely seen as important to the learning process (see, e.g., Quintana et al. 2004)—would ultimately increase time-on-task, as compared to working without feedback messages.

Increasing the amount of time students are engaged in learning has generally been shown to have a positive influence on learning outcomes (see, e.g., Nelson, 1990). This study provides further evidence of this: ChemProV's dynamic feedback system not only increased learners' time-on-task, but also improved their learning outcomes. In addition, our results suggest that, when deciding when to introduce students to dynamic feedback, instructors would do well to introduce it earlier rather than later. Indeed, students who used the "feedback" version of ChemProV first retained their high level of performance when they switched to a "no feedback" version of ChemProV. In contrast, students who first used the "no feedback" version of ChemProV first improved their task performance when they switched to the "feedback" version of ChemProV in Task 2; however, their solution accuracy remained significantly lower than that of the students who used the "feedback" version of ChemProV first.

As compared to the results obtained in our first study, the results of this study were indeed encouraging. However, given the changes we made to the problem-solving task, one needs to exercise caution in attributing too much of the observed effect to ChemProV's

redesigned feedback mechanism. In transforming the task from "solve the problem" (first study) to "eliminate the feedback messages" (this study), we may have provided participants with an additional form of scaffolding that subtly transformed the task into a more manageable one for learners. Perhaps the task of eliminating feedback messages was more doable for participants than the task of solving a problem, leading to their dedicating more attention to the feedback messages themselves, and ultimately higher problem-solving accuracy.

As was the case in our first study, we were able to use the activity logs generated by ChemProV to explore participants' actual use of feedback messages in the problem-solving process. In fact, in this study, we had access to even more log data than in the first study, since both the "feedback" and "no feedback" versions of ChemProV generated such logs. In the "no feedback" version, the messages were generated but not presented to participants (they were generated only for logging purposes), whereas in the "feedback" version, the messages were both generated and presented. Thus, we had a basis for directly assessing the impact of (visible) dynamic feedback messages on participant behavior.

On average, the "feedback" version of ChemProV generated 42.63 messages (SD = 15.14) per problem-solving session across both conditions, whereas the "no feedback" version of ChemProV generated 32.91 messages (SD = 10.13) per problem solving session. According to a two-tailed t-test with Welch's correction for unequal variances, this difference was statistically significant, $t(55.60) = 3.02$, $p = 0.004$. In other words, participants' ability to see the dynamic feedback messages significantly impacted the number of messages that were generated. We believe this was the case because, in response to the feedback messages, participants iteratively refined their solutions. Thus, they modified their solutions more often than they did without the feedback messages, and this led to the generation of even more feedback messages.

Next, we consider the percentage of messages read by participants. Recall that in the previous study, participants, on average, explicitly read just 15.75 percent of the messages presented to them. According to log data from this study, participants explicitly read 18.44 percent (*SD* = 23.56%) of the messages that were presented—a marginal improvement. However, in this study, we speculate that participants actually read far more messages than this, since they were able to read messages in the Feedback Messages Window without explicitly placing the mouse over the messages. Because of this possibility, our software logs simply could not account for all the messages that participants read. Indeed, we would have needed to enlist eye tracking technology in order to obtain more accurate data.

Notably, the percentage of messages read by participants in the Feedback First condition (*M* = 24.14%, *SD* = 31.00%) was twice as high as the percentage of messages read by participants in the No Feedback First condition (*M* = 12.75%, *SD* = 10.80%); however, due to a large amount of variance in the data, the difference was not statistically significant according to a non-parametric Kruskal-Wallis test, $H(1) = 0.52$, $p = 0.47$. (We used non-parametric tests for these data because our data failed to meet assumptions of normality.)

With respect to the number of feedback messages actually addressed by participants, we first note that, across both conditions, participants addressed a higher percentage of messages when they used the "feedback" version of the tool (*M* = 96.50%, *SD* = 8.36%) than when they used the "no feedback" version of the tool (*M* = 85.89%, *SD* = 32.74%). Not surprisingly, this difference was statistically significant, $H(1) = 14.34$, $p = 0.0002$.

Given the accuracy differences we observed by condition, one might expect the Feedback First condition to have addressed a greater number of messages than the No Feedback First condition. This was in fact the case, with the Feedback First condition addressing 95.61 percent

(*SD* = 7.29%), and the No Feedback First condition addressing 86.76 percent (*SD* = 19.43%) across both tasks. According to a non-parametric Kruskal-Wallis test, this difference was statistically significant, $H(1) = 4.54$, $p = 0.03$, and helps to explain the accuracy difference between the two conditions that we observed.

We close our discussion of this study by pointing out a shortcoming in the design of both this experiment and the first experiment we presented. Due to insufficient numbers in the courses from which we recruited participants for our studies, we were unable to include additional conditions in which participants used the same tool for *both* tasks. In the terms of experimental design, our design was not "fully crossed." While we believe we included the most important and interesting tool-task combinations in our study, we admittedly would have liked to have included in both studies at least one additional "control" condition in which participants used pen-and-paper (first study) or the "No Feedback" version of ChemProV (this study) for *both* tasks. We speculate that the solution accuracy of participants in such a "control" condition would not have improved as much from Task 1 to Task 2 as that of the participants in our studies who switched in Task 2 to ChemProV (first study) or the "feedback" version of ChemProV (this study). Nonetheless, actually including such a control condition would have provided us with a better baseline for gauging the learning gains promoted by ChemProV (first study) and the "feedback" version of ChemProV (this study) over the two problem-solving tasks.

6. Summary, Implications, and Future Work

In this article, we have suggested that a key reason engineering students fail in introductory courses is their inability to develop the skills necessary to solve basic disciplinary problems. Having observed that chemical engineering students frequently suffer from misconceptions regarding the basic syntax and semantics of the disciplinary representations of

chemical engineering (process flow diagrams and material balance equations), we have presented ChemProV, a scaffolded software environment that provides dynamic feedback on students' evolving solutions based on the syntactic and semantic rules of those disciplinary representations. In an experimental evaluation of a preliminary version of ChemProV, we found that it did not improve students' solution accuracy as compared to pen-and-paper. However, software log data alerted us to the fact that participants seldom read the feedback messages generated by the software, which meant that they had little chance of actually taking advantage of the feedback. Based on this insight, we redesigned ChemProV with an improved mechanism for presenting dynamic feedback, and ran a follow-up experimental evaluation, which found that the new feedback mechanism promoted a significant 17 percentage-point accuracy advantage over a version of ChemProV without feedback, significantly higher time-on-task, and a robust transfer-of-training effect to an unscaffolded problem-solving situation involving a version of ChemProV without feedback.

Thus, the research we have presented makes two key contributions. First, based on empirical evidence of the difficulties learners have in translating engineering problems into disciplinary representations and ultimately into sets of equations, it presents a novel software scaffolding approach to address these difficulties. Second, it demonstrates that this software scaffolding approach can be leveraged to improve students' ability to solve the kinds of "engineering translation" problems that are common in many engineering disciplines.

For engineering education, we believe our research has at least two key implications. First, engineering students who are learning to solve problems that involve translations to disciplinary representations are prone to making minor errors in which basic laws (e.g., Conservation of Mass) are violated. While these errors are seemingly minor and can be easily

detected by experts, learners often do not notice them. Uncorrected errors then impede learners' ability to derive correct equations and ultimately converge upon correct solutions. Finding ways to provide students with *timely feedback* on these errors and how they can be corrected can go a long way toward improving students' problem-solving skills.

Second, one way to provide students with timely feedback is through software environments, which can effectively *scaffold* the problem-solving process by continuously checking for these kinds of errors, and by providing dynamic, contextual feedback that alerts learners of the errors and provides hints on how to correct them. Scaffolding of this form encourages learners to spend greater time-on-task, to reflect upon their work, and to construct more accurate solutions. Ultimately, problem-solving skills acquired with the help of such scaffolding can transfer to successful problem-solving when the scaffolding is removed.

Our software design approach demonstrates the potential for software scaffolding (see, e.g., Quintana et al. 2004) and "minimalist AI" techniques (Suthers 1999) to help chemical engineering students learn how to solve material balance problems. However, we believe the approach can be used as a model for designing educationally-effective problem-solving software environments for other engineering disciplines. For example, in an introductory electrical engineering course on circuits, a scaffolded software environment analogous to ChemProV could help students to construct basic circuit diagrams, and to specify accompanying sets of equations that describe the key properties of the electricity flowing through the circuits. As students work, dynamic feedback messages could alert students to syntactic and semantic errors in their diagrams and equations. Likewise, an in introductory civil or mechanical engineering course on statics, a scaffolded software environment like ChemProV could be used to help students

construct free-body diagrams and accompanying sets of equations, providing dynamic feedback on their correctness as students solve problems.

In future research, we would like to pursue three key directions:

- *Support a broader range of problems*. The current version of ChemProV aids students in solving a limited set of material balance problems: those that involve chemical processes that are continuous and steady state. These kinds of problems are typically the focus of a few weeks of an introductory chemical engineering course. Using the same iterative, learner-centered design process (Soloway et al. 1996) we used to build the versions of ChemProV presented here, we would like to develop and empirically evaluate a new version of ChemProV that supports a broader range of problems involving both material and *energy* balances. This will allow ChemProV to be used as a learning tool throughout much of the curriculum of an introductory material/energy balance course.
- *Explore a studio-based pedagogical approach*. In the experimental studies we have presented here, individual students used ChemProV to solve problems in a closed laboratory environment. While this pedagogical approach provided ideal conditions for testing the educational effectiveness of ChemProV in a laboratory experiment, ChemProV could clearly have a greater educational impact if it were fully integrated into the curriculum of an introductory material/balance course. To that end, building on our prior research (Hundhausen, Narayanan, and Crosby 2008), we would like to explore the potential for ChemProV to be used as a foundation for a *studio-based* instructional approach similar to that commonly used in architecture and fine arts education. Such a studio-based" curriculum would revolve around learning activities in which students (a) collaboratively solve problems using the ChemProV software; (b) present their problem

solutions either in face-to-face review sessions (so called “design crits”), or asynchronously through an online system; (c) critically review each other's work; and (d) revise and improve their solutions by responding to each others' critiques. To support online peer review, we are presently integrating ChemProV into an existing online peer review environment we have developed for computing education (Hundhausen, Agrawal, and Ryan 2010). Once we have integrated ChemProV into a studio-based curriculum for an entire material/energy balance course, we will be able to conduct longer-term quasi-experimental studies that compare its effectiveness against that of a traditional instructional approach.

- *Apply software design approach in new engineering disciplines*. We believe that a key contribution of the research presented here is its potential to serve as a model for designing effective scaffolded problem-solving environments for other engineering disciplines. In future work, we would like to collaborate with educators in other engineering disciplines in the development of new scaffolded software environments modeled after ChemProV for their introductory (gateway) courses. We will then be able to test more broadly our hypothesis that the use of dynamic feedback messages based on the syntactic and semantic correctness of engineering students’ evolving problem solutions is an educationally-effective form of software scaffolding.

Acknowledgments

This research was funded by the National Science Foundation under grant no. DUE-0837828. Sterling McPherson of the Department of Psychology at Washington State University assisted us with the statistical analysis of our data. We gratefully acknowledge the assistance of Dr. Bernie Van Wie, who allowed us to conduct the first study within the context of his Spring,

2009 introductory chemical engineering class, as well as the students in the class for their participation in the study. In addition, we gratefully acknowledge the students in the Fall, 2009 material/energy balance class who participated in the second study.

References

Agarwal, P. (2009). *The design and empirical evaluation of a chemical process visualization tool to help introductory chemical engineering students solve material balance problems* (Unpublished M.S. Thesis). Washington State University.

Aleven, V., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. (2003). Help seeking and help design in interactive learning environments. *Review of Educational Research, 73*(3), 277-320.

Anderson, J., Boyle, C., & Reiser, B. (1985). Intelligent tutoring systems. *Science, 228*, 456-467.

Anderson, J., Conrad, F., & Corbett, A. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13*, 467-506.

Anderson, J., Corbett, A., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*, 167-207.

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In *Proc. 2005 ACM International Computing Education Research Workshop* (pp. 81-86). New York: ACM Press.

Besana, G., & Dettori, L. (2004). Together is better: strengthening the confidence of women in computer science via a learning community. *Journal of Computing Sciences in Colleges, 19*(5), 130-139.

Borrego, M. J., Padilla, M. A., Zhang, G., Ohland, M. W., & Anderson, T. J. (2005). Graduation rates, grade-point average, and changes of major of female and minority students entering

engineering. In *IEEE Frontiers in Education 2005* (Vol. 35, pp. T3D-1). Piscataway, NJ: IEEE.

Bransford, J., Brown, A., & Cocking, R. (Eds.). (1999). *How people learn: Brain, mind, experience, and school*. Washington, D.C.: National Academy Press.

Chi, M., Glaser, R., & Rees, E. (1982). Expertise in problem solving. In *Advances in the psychology of human intelligence* (Vol. 1). Hillsdale, NJ: Erlbaum.

Clough, D. E. (2000). Using process simulators with dynamics/control capabilities to teach unit and plantwide control strategies. In *Proc. of the AIChE Annual Meeting*. Los Angeles.

Collins, A., Brown, J., & Newman, S. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. Resnick (Ed.), *Cognition and instruction: Issues and agendas* (pp. 453-494). Mahway, NJ: Lawrence Erlbaum Associates.

Custer, R., & Daugherty, J. (2009). Professional development for teachers of engineering: Research and related activities. *The Bridge*, *39*(3).

Dahm, K. D., Hesketh, R. P., & Savelski, M. J. (2002). Is process simulation used effectively in ChE courses? *Chemical Engineering Education*, *36*(3), 192–197.

Douglas, S., Peterson, N., & Udovic, D. (1998). The Cardio-Vascular Construction Kit: Software, student laboratory manual and user's manual. In *The BIOQUEST Library CD-ROM*. San Diego: Academic Press.

Felder, R. (1986). *Elementary Principles of Chemical Processes*. New York: Wiley.

Felder, R., & Spurlin, J. (2005). Applications, reliability, and validiting of the index of learning styles. *Int. Journal of Engineering Education*, *21*(1), 103-112.

Gainen, J. (1995). Barriers to success in quantitative gatekeeper courses. *New Directions for Teaching and Learning*, *61*, 5-14. doi:10.1002/tl.37219956104

Gick, M. L. (1986). Problem-solving strategies. *Educational Psychologist, 21*(1), 99 - 120.

Guzdial, M. (1994). Software-realized scaffolding to facilitate programming for science learning. *Interactive learning Environments, 4*(1), 1-44.

Hundhausen, C., Agrawal, A., & Ryan, K. (2010). The design of an online environment to support pedagogical code reviews. In *Proc. 41st ACM technical symposium on computer science education*. New York: ACM Press.

Hundhausen, C., & Brown, J. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: an empirical study. *Computers & Education, 50*(1), 301-326.

Hundhausen, C., Narayanan, N., & Crosby, M. (2008). Exploring studio-based instructional models for computing education. In *Proc. 39th SIGCSE Technical Symposium on Computer Science Education* (pp. 392-396). New York: ACM Press.

Jonassen, D., & Land, S. (Eds.). (2000). *Theoretical Foundations of Learning Environments*. Mahwah, NJ: Lawrence Erlbaum. Retrieved from http://books.google.com/books?id=QhbBLtPudScC&pg=PA12&lpg=PA12&dq=Scaffoled+Learning+Environments&source=bl&ots=p-eN9Xu1iC&sig=b6dTSKJJOV0UvOWOXf4O5pPTkgI&hl=en&ei=mG0AS-XDGJHysQO01emHCw&sa=X&oi=book_result&ct=result&resnum=4&ved=0CB4Q6AEwAw#v=onepage&q=&f=false

Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1995). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8*, 30-43.

Larkin, J., McDermott, J., Simon, D., & Simon, H. (1980). Expert and novice performance in solving physics problems. *Science, 208*, 1335-1342.

Lave, J., & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. New

York: Cambridge University Press.

Luchini, K., Quintana, C., & Soloway, E. (2003). Pocket PiCoMap: a case study in designing and assessing a handheld concept mapping tool for learners. In *Proc. SIGCHI conference on Human factors in computing systems* (pp. 321–328). New York: ACM. Retrieved from http://doi.acm.org/10.1145/642611.642668

Metcalf, S., Krajcik, J., & Soloway, E. (2000). Model-It: A desgn retrospective. In M. Jacobson & R. Kozma (Eds.), *Innovations in science and mathematics education: Advanced designs for technologies for learning* (pp. 77-115). Mahwah, NJ: Lawrence Erlbaum Associates.

Michalchik, V., Rosenquist, A., Kozma, R., Kreikemeier, P., Schank, P., & Coppola, B. (2008). Representational esources for constructing shared understandings in the high school chemistry classroom. In J. Gilbert, M. Nakhleh, & M. Reiner (Eds.), *Visualization: Theory and practice in science education* (pp. 233-282). New York: Springer. Retrieved from http://chemsense.org/about/papers/RepresentationalResourcesMichalchik.pdf

Nelson, S. (1990). *Instructional Time as a Factor in Increasing Student Achievement*. Portland, OR: Northwest Regional Educational Lab. Retrieved from http://www.eric.ed.gov/ERICWebPortal/contentdelivery/servlet/ERICServlet?accno=ED327350

PRO/II. (n.d.). . Retrieved from http://www.ips.invensys.com/en/products/processdesign/Pages/Pro-II-P004.aspx

Quintana, C., Krajcik, J., & Soloway, E. (2002). A case study to distill structural scaffolding guidelines for scaffolded software environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing*

ourselves (pp. 81-88). New York: ACM.

Quintana, C., Reiser, B., Davis, E., Krajcik, J., Fretz, E., Duncan, R., Kyza, E., et al. (2004). A scaffolding design framework for software to support science inquiry. *Journal of the Learning Sciences*, *13*(3), 337-386.

Renkl, A., & Atkinson, R. (2003). Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective. *Educational Psychologist*, *38*(1), 15-22.

Renkl, A., & Atkinson, R. (2007). An example order for cognitive skill acquisition. In *In order to learn: How the sequence of topics influences learning* (pp. 95-106). New York: Oxford University Press.

Ritter, F., Nerb, J., Lehtinen, E., & O'Shea, T. (Eds.). (2007). *In order to learn: How the sequence of topics influences learning*. New York: Oxford University Press.

Roschelle, J. (1996). Designing for cognitive communication: Epistemic fidelity or mediating collaborative inquiry? In D. Day & D. Kovacs (Eds.), *Computers, Communication and Mental Models* (pp. 13-25). London: Taylor & Francis.

Roschelle, J., & Stroup, W. (2000). SimCalc: Accelerating students' engagement with the mathematics of change. In M. Jacobson & R. Kozma (Eds.), *Innovations in science and mathematics education: Advanced designs for technologies for learning* (pp. 47-76). Mahway, NJ: Lawrence Erlbaum Associates.

Schwonke, R., Wittwer, J., Aleven, V., Salden, R., Krieg, C., & Renkl, A. (2007). Can tutored problem solving benefit from faded worked-out examples? Presented at the European Cognitive Science conference 2007, Delphi, Greece.

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, *78*(1), 153–

189. doi:10.3102/0034654307313795

Singley, M., & Anderson, J. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.

Soloway, E., Jackson, S., Klein, J., Quintana, C., Reed, J., Spitulnik, J., Stratford, S., et al. (1996). Learning theory in practice: Case studies of learner-centered design. In *Procedings of the 1996 SIGCHI Conference on Human Factors in Computing Systems* (pp. 189-196). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/238386.238476

Superfine, A. C., Canty, R. S., & Marshall, A. M. (2009). Translation between external representation systems in mathematics: All-or-none or skill conglomerate? *The Journal of Mathematical Behavior*, *28*(4), 217 - 236. doi:DOI: 10.1016/j.jmathb.2009.10.002

Suthers, D., Hundhausen, C., & Girardeau, L. (2003). Comparing the roles of representations in face-to-face and online computer supported collaborative learning. *Computers & Education*, *41*(4), 335-351.

Suthers, D., Toth, E., & Weiner, A. (1997). An Integrated Approach to Implementing Collaborative Inquiry in the Classroom. In *Proceedings of Computer Supported Collaborative Learning* (pp. 272-279). Toronto: University of Toronto.

Suthers, D., & Xu, J. (2002). Kukakuka: an online environment for artifact-centered discourse. In *Proc. Eleventh World Wide Web Conference* (pp. 472-480). Honolulu: WWW2002.

Suthers, D. (1999). Representational and advisory guidance for learning: alternative roles for AI. In *Proc. IASTED International Conference on Artificial Intelligence and Soft Computing* (pp. 338-342). Honolulu: IASTED. Retrieved from http://lilt.ics.hawaii.edu/lilt/papers/1999/Suthers-Asc99.pdf

VanLehn, K., Jordan, P., Rose, C., Bhembe, D., Boettner, M., & Gaydos, A. (2002). The architecture of Why2-Atlas: a coach for qualitative physics essay writing. In S. Cerri, G. Gouardenes, & F. Paraguacu (Eds.), *Proc. 6th International Conference on Intelligent Tutoring Systems* (pp. 158-167). Berlin: Springer-Verlag.

Vygotsky, L. S. (1978). *Mind in society*. Cambridge, MA: Harvard University Press.

Zollars, R., Hundhausen, C., & Stefik, M. (2007). Visual learning in a material/energy balance class. In *Proc. 2007 American Society for Engineering Education Annual Conference & Exposition* (pp. AC 2007-1550). Washington, D.C.: ASEE.

Biographical Sketches

Christopher Hundhausen is an associate professor of computer science in the School of Electrical Engineering and Computer Science at Washington State University, and founder and director of the Human-centered Environments for Learning and Programming (HELP) Lab (http://helplab.org). He received his Ph.D. from the University of Oregon. Recipient of a National Science Foundation CAREER Award, Dr. Hundhausen pursues research in the general area of human-computer interaction, with a particular emphasis on the design and empirical evaluation of visualization, simulation, collaboration, and programming environments to enhance learning in computer science and engineering education.

Address: School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752; telephone: (+1) 509.335.4590; fax: (+1) 509.335.3818; e-mail: hundhaus@wsu.edu.

Pawan Agarwal is a doctoral student in the School of Electrical Engineering and Computer Science at Washington State University. His M.S. thesis, which he recently completed, focused on the preliminary design and evaluation of the ChemProV software discussed in this article.

Address: School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752; telephone: (+1) 509.335.6602; fax: (+1) 509.335.3818; e-mail: topagrawal@gmail.com.

Richard Zollars is a professor in, and Associate Director of, the Gene and Linda Voiland School of Chemical Engineering and Bioengineering at Washington State University. He received his Ph.D. from the University of Colorado. He has been teaching engineering for 30 years. His interests are colloidal/interfacial phenomena, reactor design and engineering education.

Address: Voiland School of Chemical Engineering and Bioengineering, Washington State University, Pullman, WA 99164-2710; telephone: (+1) 509.335.2670; fax: (+1) 509.335.4806; e-mail: rzollars@che.wsu.edu.

Adam Carter is a doctoral student in the School of Electrical Engineering and Computer Science at Washington State University. His dissertation research focuses on the design and evaluation of a novel novice programming environment for students learning C programming.

Address: School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752; telephone: (+1) 509.335.6602; fax: (+1) 509.335.3818; e-mail: cartera@wsu.edu.