J. EDUCATIONAL COMPUTING RESEARCH, Vol. 38(4) 469-509, 2008

HUMAN-IN-THE-LOOP: A FEEDBACK-DRIVEN MODEL FOR AUTHORING KNOWLEDGE-BASED INTERACTIVE LEARNING ENVIRONMENTS*

LIN QIU
State University of New York at Oswego

CHRISTOPHER K. RIESBECK
Northwestern University

ABSTRACT

While computer-based interactive learning environments can provide authentic and supportive settings for problem-based learning, they are very difficult to build. The traditional development model requires significant upfront development and results in systems that are hard to deploy and customize. In this article, we describe a feedback-driven authoring model that aims to reduce the development difficulty by including teachers in the feedback loop to complement system feedback and incrementally author the content in the learning environment during real use. We discuss the design of INDIE, a learning environment authoring toolkit, for supporting the incremental authoring model. We present empirical results obtained from the development and use of Corrosion Investigator, a learning environment delivered by INDIE, as an example to show how the incremental authoring model is implemented in educational settings.

*This work was supported primarily by the Engineering Research Centers Program of the National Science Foundation under Award Number EEC-9876363.

INTRODUCTION

Problem-based learning (Barrows, 2000; Bransford, Brown, & Cocking, 1999; Hmelo-Silver & Barrows, 2006; Mergendoller, Maxwell, & Bellisimo, 2006) is a pedagogical paradigm that centers learning around the investigation and development of solutions to complex and ill-structured authentic problems. In problem-based learning, students learn content knowledge and problem-solving skills through collaborative and self-directed learning. Instructors work as facilitators providing resources and coaching to students. Research indicates that with problem-based learning, students gain improvement in skills such as self-direction, critical thinking and reasoning, data gathering, and self-evaluation (e.g., Derry, Hmelo-Silver, Nagarajan, Chernobilsky, & Beitzel, 2006; Dochy, Segers, Van den Bossche, & Gijbels, 2003; Hmelo-Silver, Duncan, & Chinn, 2007; Koh, Khoo, Wong, & Koh, 2008; Torp & Sage, 2000). While problem-based learning offers an effective approach to improve teaching and learning, it has a number of drawbacks (Hoffman & Ritchie, 1997). For example, activities in solving realistic problems can be expensive and even dangerous. Students may easily lose their focus on the targeted subject matter. It is time consuming for teachers to deliver open-ended instruction.

To solve the above problems, interactive learn-by-doing environments have been built to support problem-based learning. For example, Alien Rescue (Liu, Williams, & Pedersen, 2002) is a learning environment where students need to find a new home in the solar system for aliens to survive. BioWorld (Lajoie, Lavigne, Guerrera, & Munsie, 2001) is a learning environment where students need to diagnose patients in a simulated hospital setting. Sickle Cell Counselor (Bell, Bareiss, & Beckwith, 1994) is a learning environment where students work as reproductive counselors advising couples on the health of their children. These learning environments engage students in authentic simulated scenarios and provide tools such as data portfolios to help students solve the challenges. They allow students to carry out activities that are not feasible in classrooms and receive just-in-time individualized feedback. They also reduce the instructor's effort in supporting student learning.

While interactive learning environments can facilitate problem-based learning, they are difficult to build and customize. In order to support computer-based feedback, educators and domain experts have to work with developers to implement all learning content and feedback in advance. This requires significant upfront development effort. After the system is deployed, it is very difficult for teachers to add or remove content.

To solve the above problems, we proposed an alternative model: feedback-driven incremental authoring model. This model includes the teacher in the feedback loop to complement system feedback and author content in the learning environment on demand to meet students' needs. This way, the system does not need to be completely implemented before deployment. Instead, it gradually migrates into a complete system at runtime.

In the following, we discuss the differences between the traditional development model and the incremental development model, the advantages of the incremental development model, and a learning environment authoring toolkit that we designed called Investigate-and-Decide (INDIE) to support the incremental development model. (We use Investigate-and-Decide to refer to a type of learning environment where the major activity is to investigate a problem and decide the cause of the problem.) We present empirical results at the end on the development and use of Corrosion Investigator, a learning environment delivered by INDIE, to show how the incremental authoring model can be implemented in educational settings.

TRADITIONAL DEVELOPMENT MODEL

Almost all existing development methods and authoring strategies for knowledge-based educational systems are based on the traditional waterfall software development model (Royce, 1970) or the Analyze, Design, Develop, Implement, and Evaluate (ADDIE) instructional system development model (Molenda, Pershing, & Reigeluth, 1996). Under these models, an interactive learning environment is created by software developers and educators working together to generate the learning content, determine the learning activity, form the assessment strategy, create the feedback, and develop the software. Sometimes, with appropriate authoring tools, teachers can create the learning environment without software developers (Murray, Blessing, & Ainsworth, 2003). After a learning environment is built, it is put into use. Feedback will be collected and returned to the developers. However, due to the delay in communication and the reduction of development effort, update to the learning environment is often difficult. Figure 1 shows the traditional development model of interactive learning environments.

Interactive learning environments created using the traditional model are usually static. Their contents do not change after deployment. The system is expected to handle all the interactions and feedback generation by itself. Students mainly interact with the learning environment. Instructors can provide supplemental materials, but have no control over the interaction between the students and the learning environment.

There are a number of problems in the traditional development model:

Authoring problems:

- It is difficult to anticipate and implement all the resources, actions, and feedback in advance. Problem-based learning encourages free exploration and open-ended inquiry. There are many paths to the solutions and multiple answers to the problem. To support such open-ended activities and implement all the materials upfront, it requires significant design, implementation, and

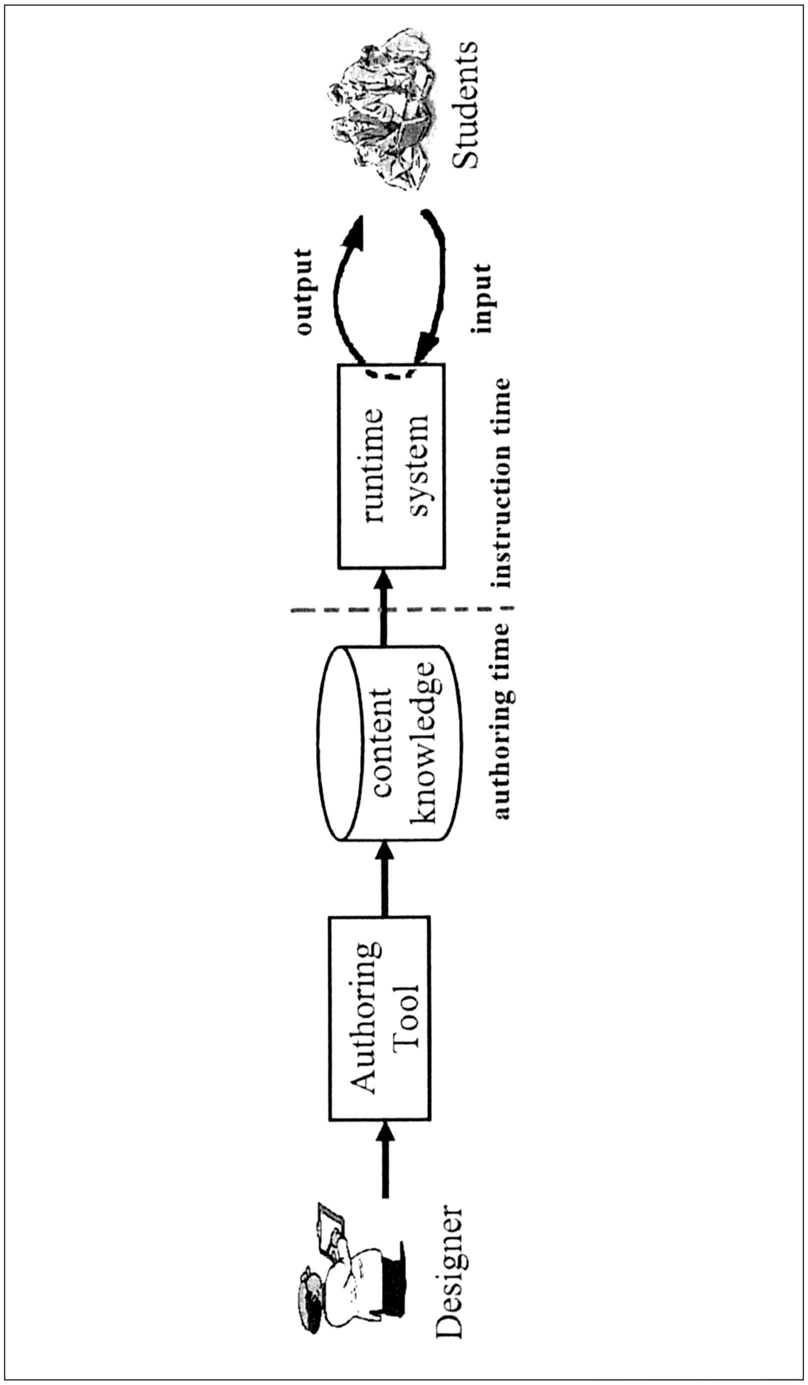


Figure 1. The traditional development model.

piloting of the system as well as expertise in the problem domain and experience with students. Even with such effort and experience, it is still likely to miss important resources and actions that students need and spend efforts on the ones that students rarely use.

- Systems that provide automatic feedback such as coaching and critiquing are very difficult to author. They must provide highly accurate feedback. Otherwise, they can easily lose credibility. Reeves and Nass (1996) shows that when users notice inappropriate feedback, they pay little attention to the feedback that is appropriate. The requirement of high accuracy significantly increases the difficulty of system development. For example, intelligent tutoring systems often require 200 to 300 hours of authoring for 1 hour of instruction (Anderson, 1993; Murray, 2003; Woolf & Cunningham, 1987). While a recent study of an authoring tool, the Cognitive Tutor Authoring Tool (CTAT), developed at the Pittsburgh Science of Learning Center, has shown to speed up the development process by 1.4 to 2 times (Aleven, McLaren, Sewall, & Koedinger, 2006), the authoring process is still considerably time-consuming.
- Systems developed using the traditional development model assume that all the necessary materials and actions have been implemented. They rarely provide tools for non-programmers to modify the learning content at instruction time. This makes it difficult for instructors to incorporate new knowledge into the system and adjust existing content for their own use.
- In the traditional development model, the benefit of using the system can only be obtained after the system is put into use. The risk of investing considerable effort at early design stages with benefits being uncertain makes instructors hesitant to participate in the system development.

Educational problems:

- Systems that are built to fully operate by themselves only allow students to do what the system has been prepared to support. Students are limited to choose existing options or paths in the system. This inevitably restricts the open-ended inquiry and free exploration encouraged by problem-based learning.
- When the learning environment handles all the feedback generation, it rarely provides interfaces for instructors to access student work in the system. Issues such as common mistakes that students make, operations that students want but cannot find, choices that students fail to investigate, and inaccurate coaching and critiquing, are hard to discover. Without knowing how students perform in the learning environment, instructors can hardly interact effectively with the students.

To solve the above problems, we proposed an alternative model: feedback-driven incremental authoring model.

FEEDBACK-DRIVEN INCREMENTAL AUTHORING MODEL

From our observations and experience, we found that the development of problem-based learning modules is an incremental process. Educators often start by choosing target content and skills, creating motivating and authentic problems, designing possible student activities, determining supporting resources, and developing evaluation strategies. Then the design is put into practice with students. Although the design was created by designers with their best knowledge, unanticipated situations often occur during practice. By handling those situations, the instructor improves his/her understanding of how students approach the problem, and continuously incorporates new materials into the teaching module. Finally, the module contains enough materials to handle most student requests and can be shared with other educators.

Based on the above analysis, we developed a feedback-driven incremental authoring model (see Figure 2) that observes the natural development process of problem-based learning modules. In this model, a learning environment is initially built with a challenge statement, relevant background information, common actions that students will take, and feedback for those actions. It does not need to have all the possible resources and feedback, but is sufficient for students to start working in the learning environment. When student inputs can be handled by the system, the system provides automatic feedback. The instructor can opt to verify and improve the feedback before it is delivered to students. When student inputs cannot be handled by the system, they are sent to the instructor. The instructor provides feedback to these inputs and, more importantly, incorporates new materials into the system, and improves the system performance on demand. This model allows the instructor to have the benefits provided by computer-based environments, and at the same time lets the system be improved by the instructor.

In the incremental authoring model, the learning environment plays two roles. On one hand, it serves as a supportive environment where students perform authentic problem-solving tasks. One the other hand, it works as a vehicle for accumulating materials for authoring. The instructor also plays two important roles. The instructor is a user who uses the learning environment to help deliver a problem-based learning module. The instructor is also an author who improves the system on demand.

The incremental authoring model has the following advantages:

- *In-context authoring:* In the incremental authoring model, authoring is done in the context of addressing students' needs. The instructor can gather materials such as students' inquiries, common mistakes, and corresponding critiques, and use them to augment the system. It is much easier than predicting what students might do beforehand.
- *Authoring driven by real needs:* In the incremental authoring model, authoring is done to meet real needs. For example, when students need more

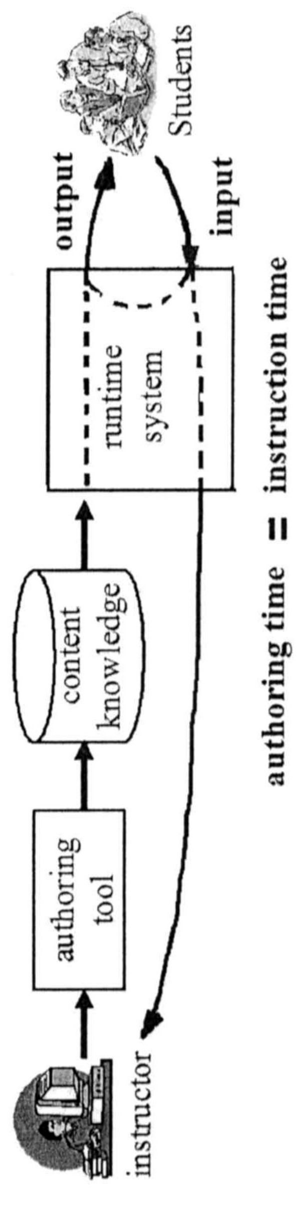


Figure 2. The feedback-driven incremental authoring model.

background information, the instructor adds new materials into the system. When students need a new test, the instructor adds the new test into the system. No authoring effort will be wasted.

- *Amortizing authoring effort:* In the incremental authoring model, materials can be added into the system gradually during use time. There is no need to anticipate and implement all possible situations upfront. Expensive knowledge engineering is not done upfront, but gradually on demand, where it is needed.
- *Early deployment:* Because there is a human in the feedback loop to complement automatic feedback, there is no need to have complete and extremely accurate system feedback. The system can be put into use when system feedback is not yet mature and reliable.
- *Extensible content:* In the incremental authoring model, issues not anticipated during system design can be explored and incorporated into the system later. This implies content repair, refinement, adaptation, and customization to different scenarios and student bodies. Such capability keeps the system from depending on predefined content after deployment.
- *In-context and real-time assessment:* Working in the feedback loop allows the instructor to have access to student learning in the system. This can provide important information for assessing how well students learn in the problem-solving process (Wiggins, 1992). The instructor can work alongside with the students to provide ongoing formative coaching and critiquing. This is considered an effective way to foster learning in the cognitive apprenticeship model (Collins, Brown, & Newman, 1989).

The incremental authoring model is similar to the Wizard of Oz approach (Wilson & Rosenberg, 1988) for prototyping systems. In the Wizard of Oz approach, a human "wizard" simulates the behavior of a system behind an interface to interact with the user. Data collected during the interaction is used to construct the system. While the incremental authoring model also has a human work behind the system to provide feedback to students, the data collection process does not occur at a separate prototyping stage but happens during the use of the system. Furthermore, the instructor does not pretend to be a computer, but works with the computer to provide feedback and perform authoring.

The incremental authoring model is an application of the agile software development methodology (Beck, 2000; Cockburn, 2002) in developing interactive learning environments. The agile software development methodology uses methods such as rapid prototyping and iterative development (Larman, 2003). It develops systems by first creating a working system and then gradually updating the system according to user needs. For example, seeding, evolutionary growth, reseeding (SER) is a model of developing software through three *evolutionary stages* (Fischer 1998; Fischer & Ostwald 2002). Seeding is the first stage where a system is created with initial knowledge that enables the system to be used for practice. Evolutionary growth is the stage where the system supports user work and collects information generated during use. Reseeding is the stage where information collected during evolutionary growth is formalized and organized to support the next cycle of development. While the incremental authoring model also uses an evolutionary approach, it does not have an explicit optimization stage. Content is incorporated into the system gradually during use. Furthermore, the incremental authoring model allows teachers to directly make changes to the learning content without the intervention of software developers. This makes the update of the system much easier.

The incremental authoring model is also an instance of user-centered design (Norman & Draper, 1986) and participatory design (Schuler & Namioka, 1993). User-centered design and participatory design encourage user participation in the development process to help developers create software that adequately addresses user needs. Likewise, the incremental authoring model allows teachers to improve the learning environment to meet students' needs. Furthermore, it makes needs collection easier by including teachers in the feedback loop to receive requests from students and critique students' work.

The incremental authoring model is similar to the design-based research paradigm (Brown, 1992; Collins, Joseph, & Bielaczyc, 2004; Edelson, 2002; Wang & Hannafin, 2005) in that they both underscore the importance of context in developing technology-enhanced learning environments. Design-based research emphasizes creating design through iterative analysis, implementation, and revision in real-world settings. The incremental authoring model emphasizes including teachers in the feedback loop to create learning content in the context of supporting student learning. The difference is that design-based research is a general framework that can be applied to the development of teaching materials such as videos, but the incremental authoring model is a framework specific for developing knowledge-based learning environments.

Like many other authoring tools such as the Cognitive Tutor Authoring Tools (Aleven et al, 2006; Koedinger, Aleven, Heffernan, McLaren, & Hockenberry, 2004), we use the component design approach (Roschelle, Kaput, Stroup, & Kahn, 1998) to divide the authoring effort. The authoring toolkit in the incremental authoring model will be built by software developers and the teachers will use the authoring toolkit to build the learning content. While the initial development of the authoring toolkit still requires considerable effort, the difficulty of developing the learning content will be reduced. Furthermore, teachers are now capable of creating learning content according to their own needs.

INDIE

INDIE is a software toolkit for authoring and delivering Web-based interactive learning environments where students need to run experiments, interpret data, generate hypotheses, and make arguments. It is based on the goal-based scenario

(GBS) framework (Schank, Fano, Bell, & Jona, 1993) and is specifically designed to create learning environments for scientific inquiry.

An old version of INDIE was developed in Lisp and only ran on Mac. Over a dozen of interactive learning environments were built with the old INDIE (Dobson, 1998). They include learning environments for diagnosing patients with nutrition-related difficulties, investigating the likelihood of volcano eruption, examining the authenticity of Rembrandtesque paintings, etc. The new INDIE uses Web-based technology for better accessibility and deployability. It includes a domain-independent runtime engine for delivering the learning environment, and an authoring tool for specifying the content in the learning environment.

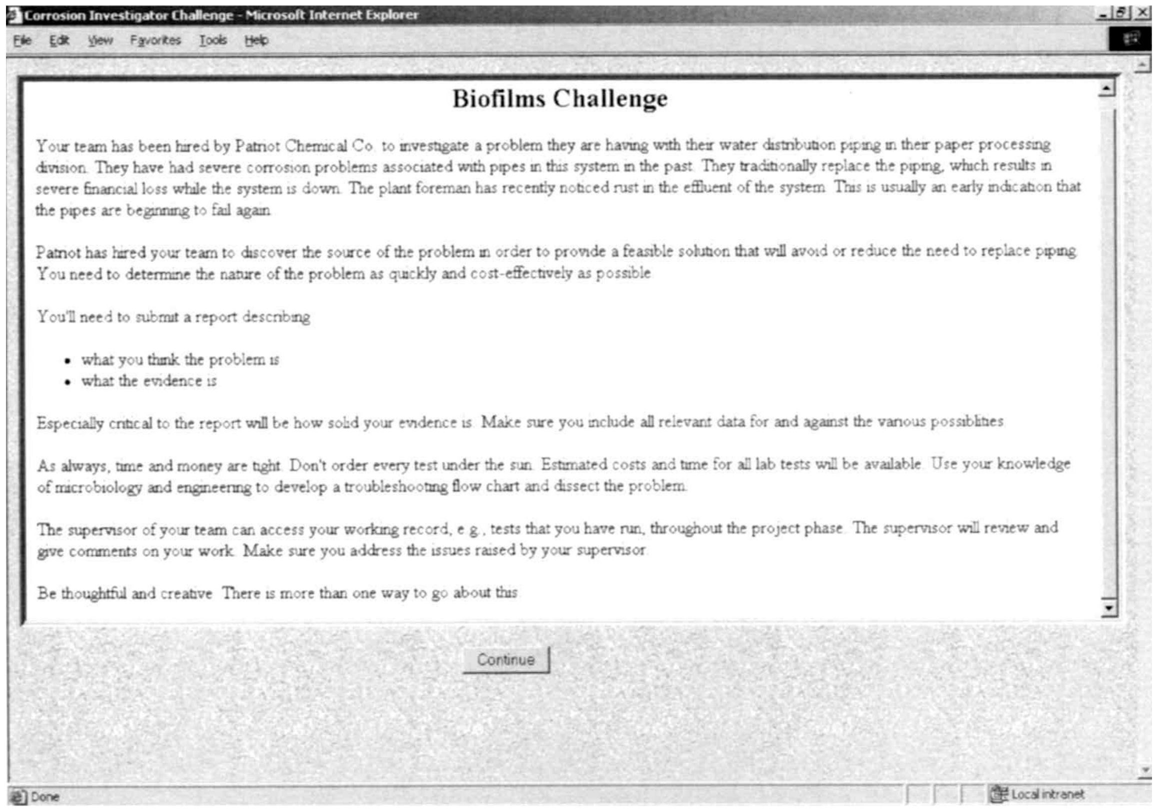
The new INDIE is different from other authoring tools in that it is for teachers to use after the learning environment is in use by the students. Murray (2003) described a number of authoring tools for learning environments. For example, SimQuest (de Jong & van Joolingen, 1998; van Joolingen & de Jong, 1996) is an authoring tool for building simulations for discovery learning. RIDES (Munro et al., 1997) is an authoring tool for interactive graphical simulations with integrated training tutorials. XAIDA (Hsieh, Halff, & Redfield, 1999) is an authoring tool for learning environments teaching device operation and maintenance. LEAP (Sparks, Dooley, Meiskey, & Blumenthal, 1998) is an authoring tool for learning environments training customer service employees how to respond to customer requests. These authoring tools including the old INDIE are all for use at design time before the learning environment is put into use. Learning environments created by these authoring tools are all closed systems. They do not allow the teacher to change the learning content. In contrast, the new INDIE is for teachers to use after the learning environment is deployed. It includes the teacher in the feedback loop to complement system feedback and author the learning content to meet students' needs.

In the following, we use Corrosion Investigator, an INDIE learning environment, as an example to show the kind of learning environments INDIE can deliver. Then, we describe how INDIE is designed to support the incremental authoring model.

Corrosion Investigator

Corrosion Investigator is a learning environment delivered by INDIE on environmental engineering. In Corrosion Investigator, students take the role of consultants helping a paper processing company find the cause of recurring pipe corrosion.

When students enter Corrosion Investigator, a *challenge* screen (Figure 3) tells students that they need to diagnose the cause of two corrosion problems occurring in the pipeline in a paper processing company and create a report with evidence supporting their diagnosis. After reading the challenge, students can go to the *reference* screen (Figure 4). This screen contains background


Corrosion Investigator Challenge - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Continue
Done
Local intranet

Biofilms Challenge

Your team has been hired by Patriot Chemical Co. to investigate a problem they are having with their water distribution piping in their paper processing division. They have had severe corrosion problems associated with pipes in this system in the past. They traditionally replace the piping, which results in severe financial loss while the system is down. The plant foreman has recently noticed rust in the effluent of the system. This is usually an early indication that the pipes are beginning to fail again.

Patriot has hired your team to discover the source of the problem in order to provide a feasible solution that will avoid or reduce the need to replace piping. You need to determine the nature of the problem as quickly and cost-effectively as possible.

You'll need to submit a report describing

- what you think the problem is
- what the evidence is

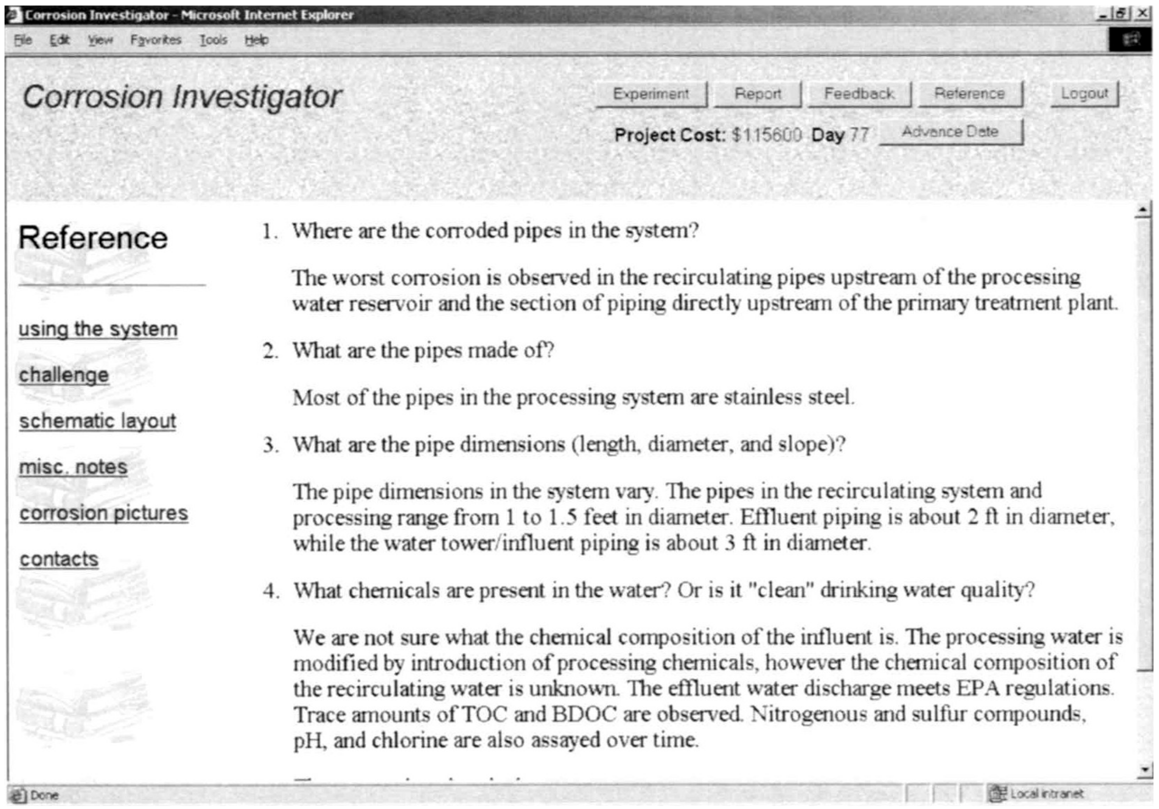
Especially critical to the report will be how solid your evidence is. Make sure you include all relevant data for and against the various possiblities.

As always, time and money are tight. Don't order every test under the sun. Estimated costs and time for all lab tests will be available. Use your knowledge of microbiology and engineering to develop a troubleshooting flow chart and dissect the problem.

The supervisor of your team can access your working record, e.g., tests that you have run, throughout the project phase. The supervisor will review and give comments on your work. Make sure you address the issues raised by your supervisor.

Be thoughtful and creative. There is more than one way to go about this.

Figure 3. The *challenge* screen in Corrosion Investigator.



Project Cost: $115600 **Day** 77

Reference

1. Where are the corroded pipes in the system?

   The worst corrosion is observed in the recirculating pipes upstream of the processing water reservoir and the section of piping directly upstream of the primary treatment plant.

2. What are the pipes made of?

   Most of the pipes in the processing system are stainless steel.

3. What are the pipe dimensions (length, diameter, and slope)?

   The pipe dimensions in the system vary. The pipes in the recirculating system and processing range from 1 to 1.5 feet in diameter. Effluent piping is about 2 ft in diameter, while the water tower/influent piping is about 3 ft in diameter.

4. What chemicals are present in the water? Or is it "clean" drinking water quality?

   We are not sure what the chemical composition of the influent is. The processing water is modified by introduction of processing chemicals, however the chemical composition of the recirculating water is unknown. The effluent water discharge meets EPA regulations. Trace amounts of TOC and BDOC are observed. Nitrogenous and sulfur compounds, pH, and chlorine are also assayed over time.

Figure 4. The *reference* screen in Corrosion Investigator.

information about the company, their pipe layout, and the location and condition of the pipe corrosion. Students can ask questions to four characters in the scenario: the plant foreman, the plant manager, the scientific consultant, and the supervisor. Questions directed to these characters are forwarded to the instructor. The instructor provides answers by taking the role of these characters. For example, when students ask the plant foreman if he smells anything from the pipe, the instructor can answer "I sometimes smell rotten-eggs."

To run tests to diagnose the corrosion problem, students go to the *experiment* screen (Figure 5). The left side of the screen has the *notebook* and *result* area. The *notebook* automatically collects all the test results that students receive from the system and splits them into single items with labels indicating their test names and conditions. It helps students keep track of all the test results received from the system. Test results in the notebook are clickable items. Students can select them to use as evidence in their reports. The *result* area displays test results in a readable form, typically a table with labeled columns and rows.

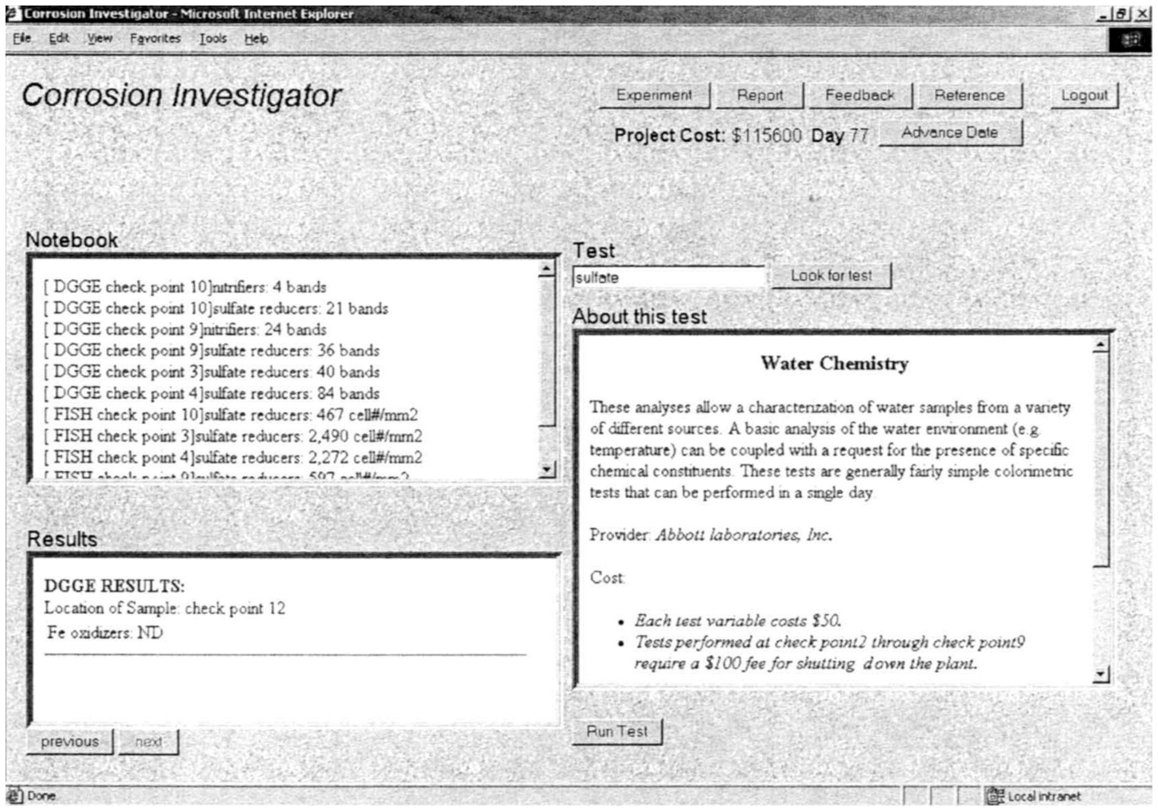
The right side of the *experiment* screen allows students to look for tests by entering test names into a textbox. Tests matching the name will be shown. Students can view the description of the tests and possible variable values for the tests. When students decide to run a test, they can specify the parameters for the test on a separate screen (Figure 6). For example, there are two parameters for the water chemistry test, Location of Sample and Test Variable, one with 12 options and one with 9 options. Tests in Corrosion Investigator often have complex test options so that students have to think hard about which tests to run.

The *cost* and *delay* field on the parameter selection screen displays the simulated amount of money and the days the test takes. These values are dynamically calculated and displayed based on the parameter selection. They will be added to the value of the *project cost* and *day* field on the top of the screen. These fields simulate that tests in real-life cost time and money. They prompt students to solve the challenge using minimum cost and allow the teacher to evaluate student learning based on how much time and money they spend.

In addition to selecting values for test parameters, students also need to enter reasons for ordering the test. This allows teachers to evaluate students' understanding of tests and their problem-solving strategy.

To receive test results, students need to press the *advance date* button at the top of the screen to advance the simulated project date to the time when the most recent test results are available. Newly available test results will appear in both the *notebook* and *result* area on the *experiment* screen.

When students feel they have gathered enough information, they can go to the *report* screen (Figure 7) to use test results in the *notebook* as evidence to support their claims. Students can pick a corrosion location and enter their diagnosis. When they select a result in the *notebook*, a window will pop up allowing them to enter the reason for using the test result as evidence. The report



Project Cost: $115600 **Day** 77

Test: sulfate

[DGGE check point 10]nitrifiers: 4 bands
[DGGE check point 10]sulfate reducers: 21 bands
[DGGE check point 9]nitrifiers: 24 bands
[DGGE check point 9]sulfate reducers: 36 bands
[DGGE check point 3]sulfate reducers: 40 bands
[DGGE check point 4]sulfate reducers: 84 bands
[FISH check point 10]sulfate reducers: 467 cell#/mm2
[FISH check point 3]sulfate reducers: 2,490 cell#/mm2
[FISH check point 4]sulfate reducers: 2,272 cell#/mm2

Water Chemistry

These analyses allow a characterization of water samples from a variety of different sources. A basic analysis of the water environment (e.g. temperature) can be coupled with a request for the presence of specific chemical constituents. These tests are generally fairly simple colorimetric tests that can be performed in a single day.

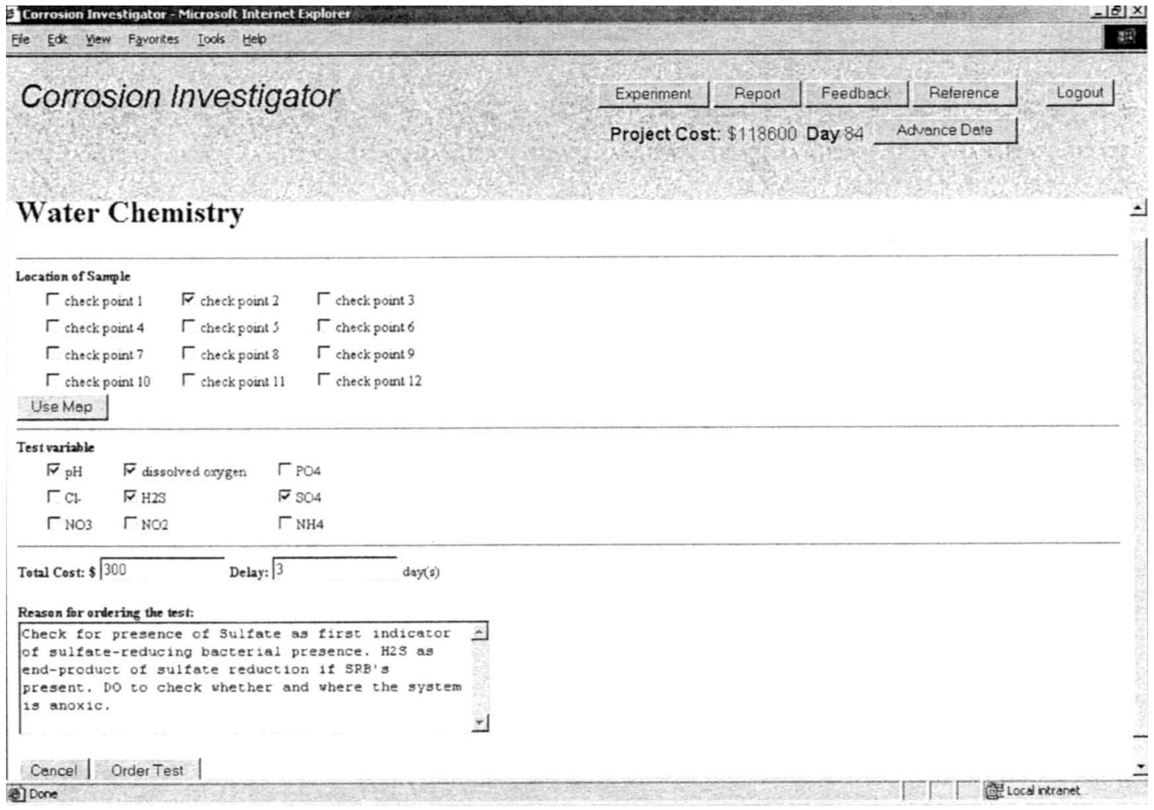
Provider: *Abbott laboratories, Inc.*

Cost:

- *Each test variable costs $50.*
- *Tests performed at check point2 through check point9 require a $100 fee for shutting down the plant.*

DGGE RESULTS:
Location of Sample: check point 12
Fe oxidizers: ND

Figure 5. The *experiment* screen in Corrosion Investigator.



Project Cost: $118600 **Day** 84

Water Chemistry

Location of Sample

☐ check point 1 [✓ check point 2] ☐ check point 3
☐ check point 4 ☐ check point 5 ☐ check point 6
☐ check point 7 ☐ check point 8 ☐ check point 9
☐ check point 10 ☐ check point 11 ☐ check point 12

Test variable

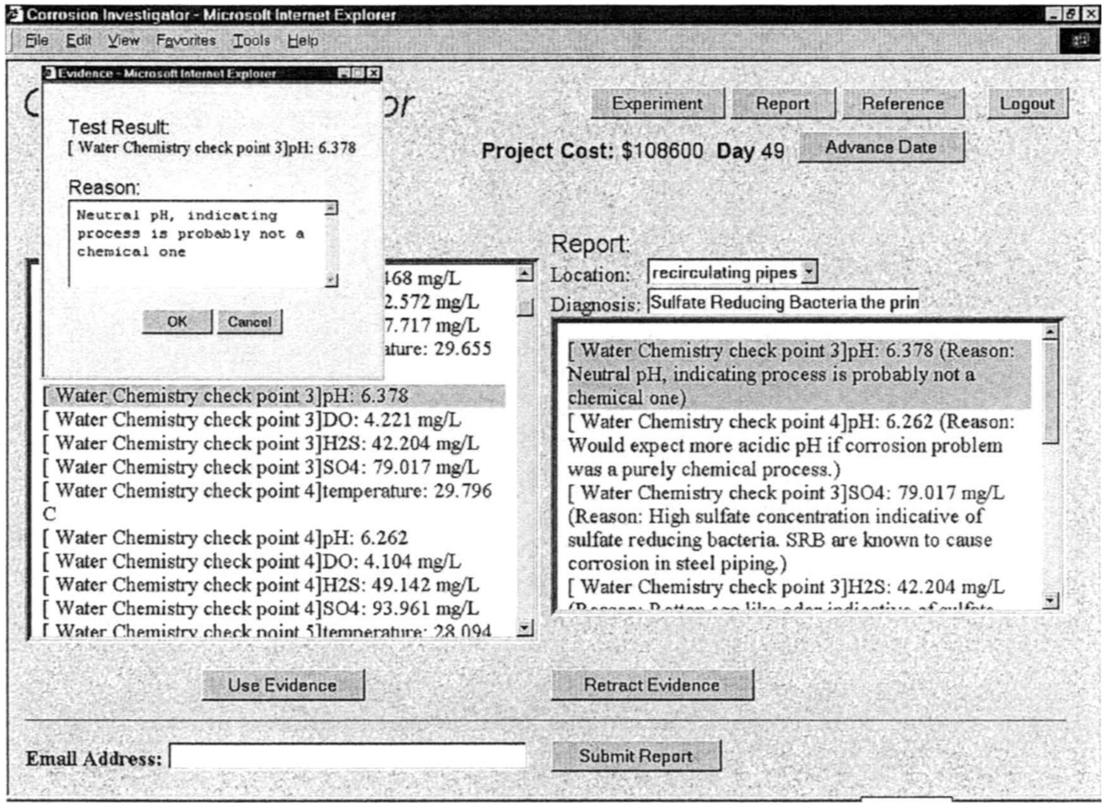
[✓ pH] [✓ dissolved oxygen] ☐ PO4
☐ Cl- [✓ H2S] [✓ SO4]
☐ NO3 ☐ NO2 ☐ NH4

Total Cost: $ 300 **Delay:** 3 day(s)

Reason for ordering the test:

```
Check for presence of Sulfate as first indicator
of sulfate-reducing bacterial presence. H2S as
end-product of sulfate reduction if SRB's
present. DO to check whether and where the system
is anoxic.
```

Figure 6. The parameter value selection screen in Corrosion Investigator.


Corrosion Investigator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Evidence - Microsoft Internet Explorer
Experiment
Report
Reference
Logout
Advance Date
OK
Cancel
Use Evidence
Retract Evidence
Submit Report

Test Result:
[Water Chemistry check point 3]pH: 6.378

Reason:

```
Neutral pH, indicating
process is probably not a
chemical one
```

Project Cost: $108600 Day 49

468 mg/L
2.572 mg/L
7.717 mg/L
ature: 29.655
[Water Chemistry check point 3]pH: 6.378
[Water Chemistry check point 3]DO: 4.221 mg/L
[Water Chemistry check point 3]H2S: 42.204 mg/L
[Water Chemistry check point 3]SO4: 79.017 mg/L
[Water Chemistry check point 4]temperature: 29.796 C
[Water Chemistry check point 4]pH: 6.262
[Water Chemistry check point 4]DO: 4.104 mg/L
[Water Chemistry check point 4]H2S: 49.142 mg/L
[Water Chemistry check point 4]SO4: 93.961 mg/L
[Water Chemistry check point 5]temperature: 28.094

Report:
Location: recirculating pipes
Diagnosis: Sulfate Reducing Bacteria the prin

[Water Chemistry check point 3]pH: 6.378 (Reason: Neutral pH, indicating process is probably not a chemical one)
[Water Chemistry check point 4]pH: 6.262 (Reason: Would expect more acidic pH if corrosion problem was a purely chemical process.)
[Water Chemistry check point 3]SO4: 79.017 mg/L (Reason: High sulfate concentration indicative of sulfate reducing bacteria. SRB are known to cause corrosion in steel piping.)
[Water Chemistry check point 3]H2S: 42.204 mg/L

Email Address:

Figure 7. The *report* screen in Corrosion Investigator.

will be evaluated by the instructor in terms of the correctness of their diagnoses and the relevance of the evidence to their diagnoses.

While students are working in the system, their work is under review by their supervisor (role-played by the instructor). The supervisor can add comments to the students' work. Students can review these comments on the *feedback* screen (Figure 8) and respond by clicking the *respond* link and enter their responses in a pop-up window.

DESIGN FOR INCREMENTAL AUTHORING

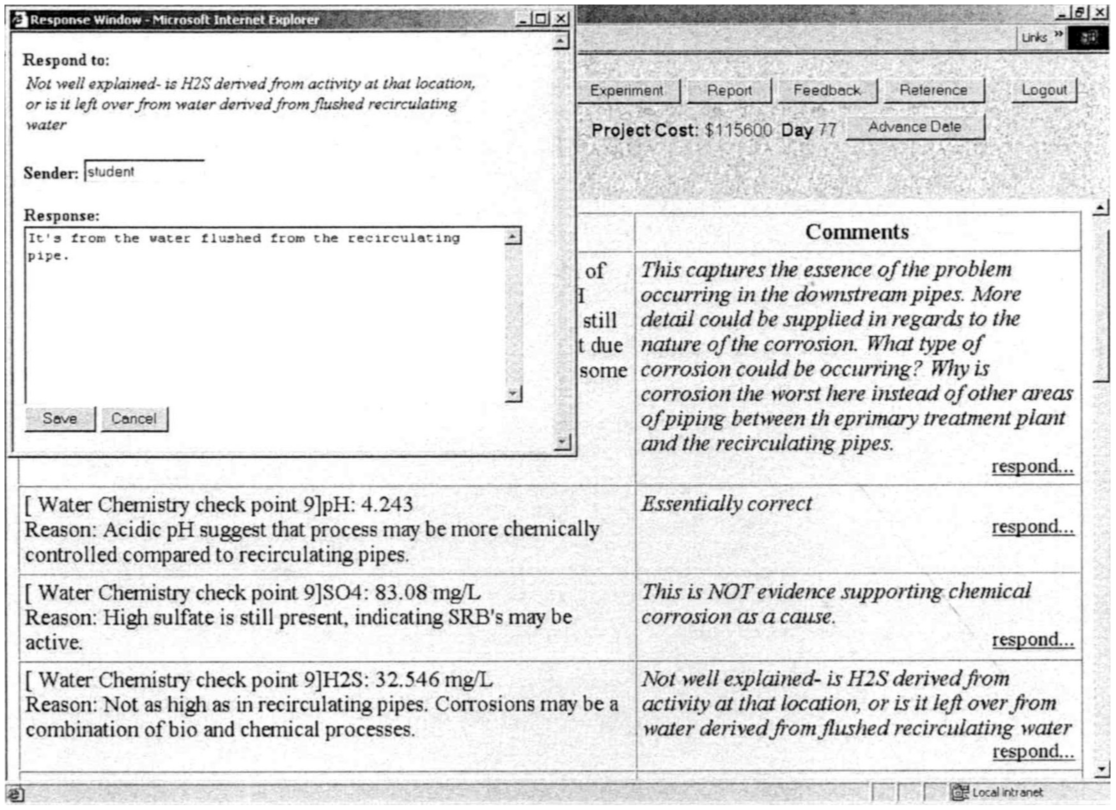
Developing a learning environment like Corrosion Investigator requires significant software development effort as well as considerable expertise in the subject domain. After the learning environment is put into use, it is still likely that important learning content may be missing. For example, students may want to run a particular test that was not expected by the domain experts. To solve this problem, we developed INDIE to support incremental authoring of the content in learning environments like Corrosion Investigator. In the following, we describe how INDIE is designed to support the four key elements in the incremental authoring model:

1. allowing teachers to author learning content without programming;
2. allowing teachers to author learning content at runtime;
3. allowing teachers to complement system feedback; and
4. allowing teachers to collect materials for authoring.

General Interface Framework with an Authoring Tool

The incremental authoring model requires teachers to author content in the learning environment at runtime. This requires the authoring task and involves minimum programming. In INDIE, we use a general interface framework with an authoring tool to allow teachers to author the learning content without programming. The framework consists of a set of Web interfaces: the *challenge* screen, the *reference* screen, the *experiment* screen, the *report* screen, and the *feedback* screen (as shown in the Corrosion Investigator section). This framework includes important learning tools such as the persistent structured portfolio (the *notebook*) and the argument construction tool (the *report*). Teachers can specify all contents using static data. They do not need to write rules or scripts to handle student interactions.

Learning content in INDIE learning environments consists of scenario information such as the challenge statement, and test information such as the cost and delay of a test and test result generation methods. The scenario information is described in Webpages which can be easily authored using any off-the-shelf Webpage authoring tools and uploaded into the learning environment. Information about tests can be authored through a form-based interface provided by the



Respond to:
Not well explained- is H2S derived from activity at that location, or is it left over from water derived from flushed recirculating water

Sender: student

Response:
```
It's from the water flushed from the recirculating
pipe.
```

Project Cost: $115600 **Day** 77

	Comments
of I still t due some	*This captures the essence of the problem occurring in the downstream pipes. More detail could be supplied in regards to the nature of the corrosion. What type of corrosion could be occurring? Why is corrosion the worst here instead of other areas of piping between th eprimary treatment plant and the recirculating pipes.* respond...
[Water Chemistry check point 9]pH: 4.243 Reason: Acidic pH suggest that process may be more chemically controlled compared to recirculating pipes.	*Essentially correct* respond...
[Water Chemistry check point 9]SO4: 83.08 mg/L Reason: High sulfate is still present, indicating SRB's may be active.	*This is NOT evidence supporting chemical corrosion as a cause.* respond...
[Water Chemistry check point 9]H2S: 32.546 mg/L Reason: Not as high as in recirculating pipes. Corrosions may be a combination of bio and chemical processes.	*Not well explained- is H2S derived from activity at that location, or is it left over from water derived from flushed recirculating water* respond...

Figure 8. The *feedback* screen in Corrosion Investigator.

INDIE authoring tool. We use the Naked Objects approach (Pawson & Mathews, 2002) to generate the interface. We use a generic engine to query the attributes (i.e., data structures) of each test and create the corresponding authoring interface. For example, for a primitive type (i.e., int, double, etc.) or String type attribute (e.g., the cost of a test), a text field will be created. For a Boolean type attribute, a menu with two options, true and false, will be created. For any attribute whose value is a list of items, a list box will be created containing all the items. For any attribute that does not belong to the above types, a hyperlink will be created leading to another screen for editing the attribute. The above mapping generates a uniform authoring interface that allows authors to easily master the use of the interface after an initial learning process. For example, Figure 9 shows the interface for authoring a test (the culturing test in Corrosion Investigator). Values for primitive type attributes (e.g., the name and the cost) can be specified in textboxes. Attributes that have a list of items as their values (such as the parameters for the test) are displayed in a list box. Items in the list can be selected and edited in another screen in the same format. For example, Figure 10 shows the interface for authoring a new parameter for the culturing test. Authors can specify the cost and delay of the parameter, and the test options for the parameter.

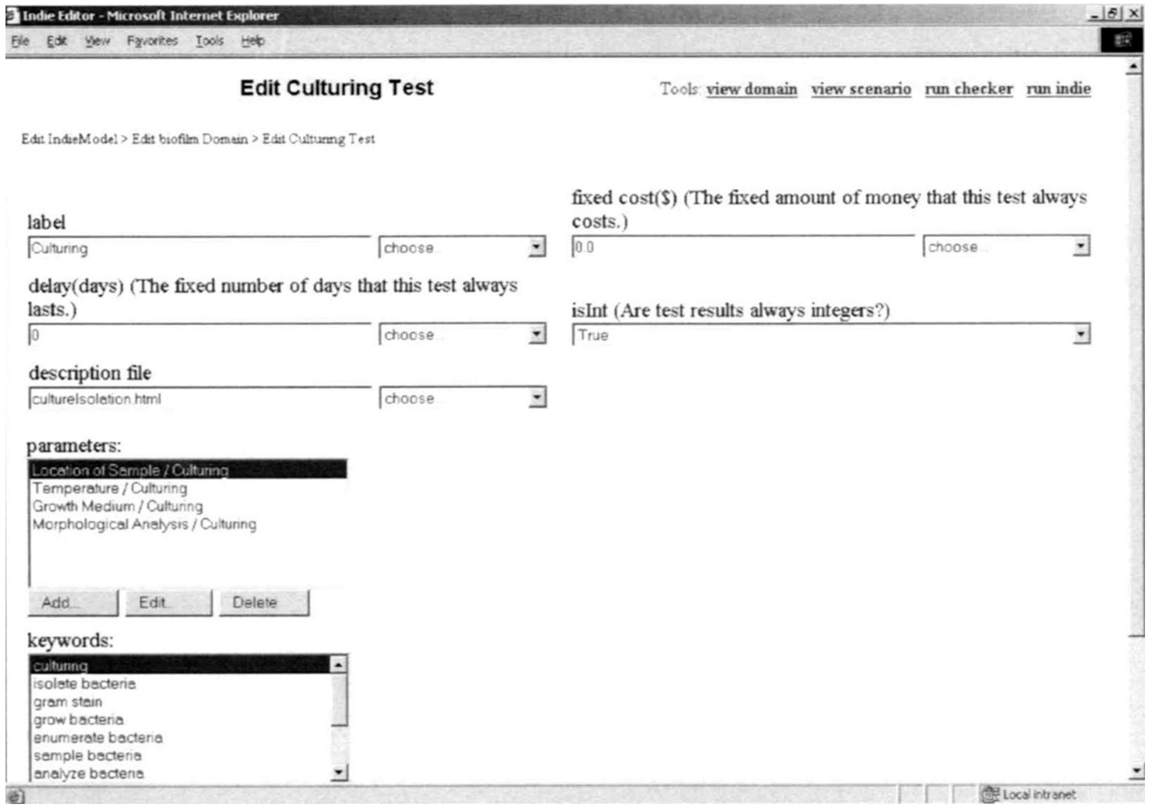
The INDIE authoring tools further provides a list of features such as preview of student interface, overview of learning content, type check, and completeness check to facilitate authoring. For details, see Qiu (2005).

Web-Based Client-Server Architecture

The runtime incremental authoring model requires the teacher to receive requests from students and constantly provide feedback and author learning content. This was not possible when educational software needed to be installed individually on each student's machine. To update the learning content, the teacher would have to update the software on each student's machine.

Now, with the widespread use of Web technology, software no longer needs to be installed on individual machines. It can be accessed anywhere anytime from Web browsers. This makes the deployment and update of software significantly easier.

We take advantage of the Web technology in INDIE to support runtime incremental authoring. INDIE saves all the learning content on a centralized server and provides a Web-based authoring tool for teachers to modify the content. With the Web-based authoring tool, teachers can modify the learning content anytime anywhere. INDIE further provides a Web interface for students to interact with the learning environment. This enables any update of the learning content immediately and reflects in the student's learning environment. Students can benefit from the most recent content authored by the teacher. The Web-based architecture also saves students' work on the server so that teachers can easily


Indie Editor - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Add... Edit... Delete
Local intranet

Edit Culturing Test

Tools: view domain view scenario run checker run indie

Edit IndieModel > Edit biofilm Domain > Edit Culturing Test

label
Culturing | choose...

delay(days) (The fixed number of days that this test always lasts.)
0 | choose...

description file
cultureIsolation.html | choose...

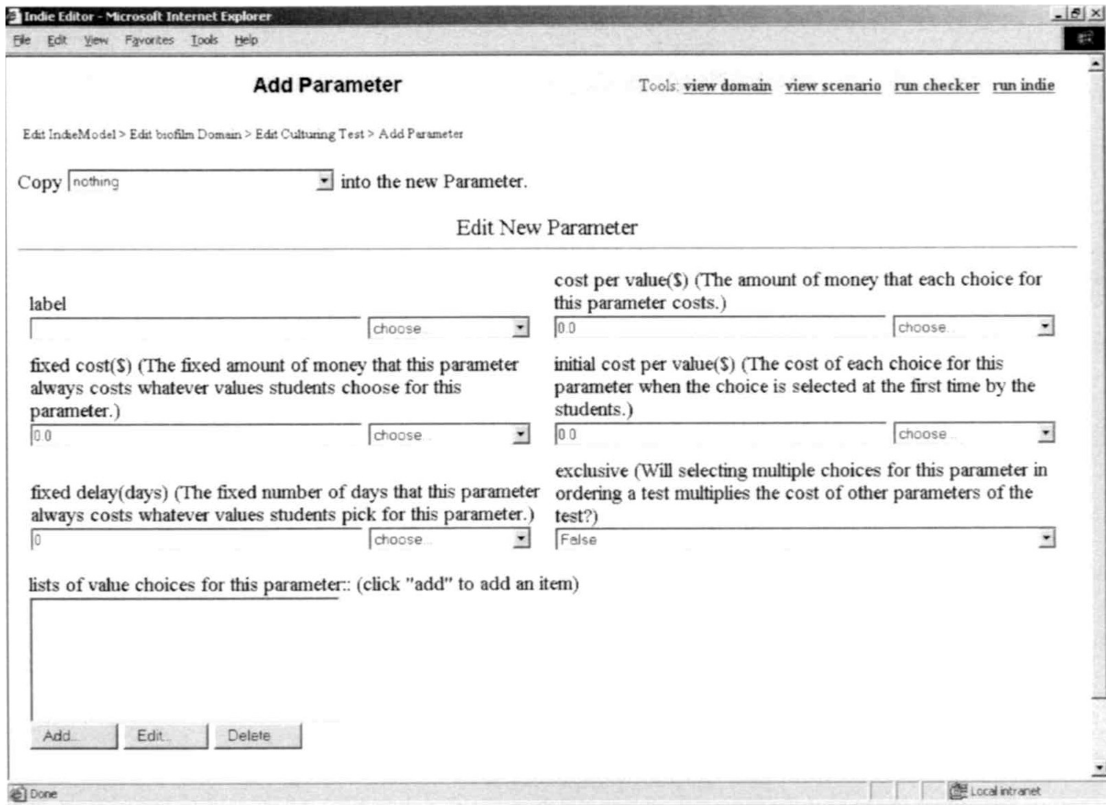
fixed cost($) (The fixed amount of money that this test always costs.)
0.0 | choose...

isInt (Are test results always integers?)
True

parameters:
Location of Sample / Culturing
Temperature / Culturing
Growth Medium / Culturing
Morphological Analysis / Culturing

keywords:
culturing
isolate bacteria
gram stain
grow bacteria
enumerate bacteria
sample bacteria
analyze bacteria

Figure 9. The interface for authoring a test.


Indie Editor - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Add Parameter

Tools: view domain view scenario run checker run indie

Edit IndieModel > Edit biofilm Domain > Edit Culturing Test > Add Parameter

Copy nothing into the new Parameter.

Edit New Parameter

label

choose

cost per value($) (The amount of money that each choice for this parameter costs.)

0.0 choose

fixed cost($) (The fixed amount of money that this parameter always costs whatever values students choose for this parameter.)

0.0 choose

initial cost per value($) (The cost of each choice for this parameter when the choice is selected at the first time by the students.)

0.0 choose

fixed delay(days) (The fixed number of days that this parameter always costs whatever values students pick for this parameter.)

0 choose

exclusive (Will selecting multiple choices for this parameter in ordering a test multiplies the cost of other parameters of the test?)

False

lists of value choices for this parameter:: (click "add" to add an item)

Figure 10. The interface for adding a new parameter to a test.

review it through Web interfaces. Teachers no longer need to collect student records from individual machines. This makes it possible to include the teacher in the feedback loop to provide feedback to students' work in the learning environment.

Human-in-the-Loop Hybrid Feedback Generation Mechanism

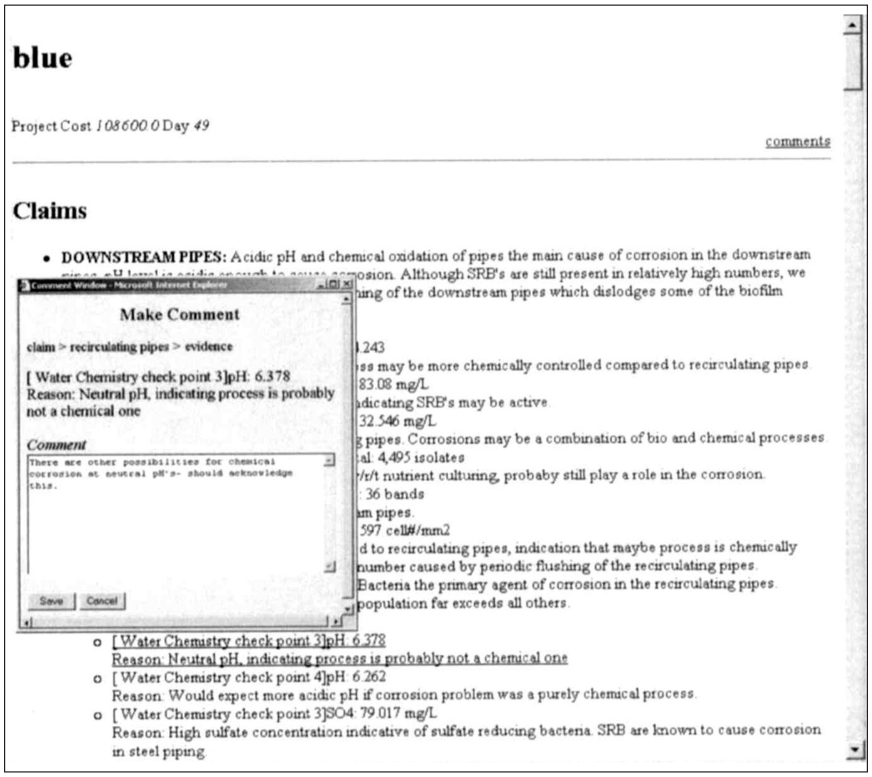
In the incremental authoring model, a learning environment is put into use while it does not have all the actions and feedback that students need. This requires an infrastructure to allow the teacher to help the system meet students' needs. (This actually also provides a great opportunity for the teachers to learn what is needed by the students and what should be added into the learning environment.)

We use a hybrid feedback generation mechanism in INDIE to include the teacher in the feedback loop to complement the system feedback. In INDIE, the system is responsible for generating feedback that is immediately required or computational-intensive. For example, it generates the cost and time of a test and complicated test results. The teacher is responsible for generating feedback that requires natural language understanding and expert knowledge. For example, the teacher provides answers to questions about the scenario and critiques students' work.

To allow teachers to receive requests from students, INDIE forwards students' requests to virtual characters in the scenario to the teacher. For example, in Corrosion Investigator, students can ask questions to characters in the scenario (e.g., the plant foreman and lab manager). Questions sent to these characters are forwarded to the teacher. This allows the teacher to provide extra information (such as background information and test results) to students in an authentic problem context.

For critiquing, INDIE organizes and displays students' work in the learning environment in an interactive report (see Figure 11) for teachers to review. The report includes the time and money that students have spent, tests that students have scheduled and run, reasons for running those tests, and diagnoses and supporting evidence that students have created. Items in the report are clickable links. Teachers can click on any of them and a pop-window will allow the teacher to enter critiques. These critiques will appear in the *feedback* screen in the learning environment. The interactive report is automatically updated every time when the students make a move in the learning environment so that the teacher always sees the most recent student activity.

The above hybrid feedback generation mechanism provides a natural human-computer integration where the system takes care of repetitive and well-defined feedback generation, and the human takes care of the work that is open-ended and requires human intelligence.



blue

Project Cost *108600.0* Day *49*

comments

Claims

- **DOWNSTREAM PIPES:** Acidic pH and chemical oxidation of pipes the main cause of corrosion in the downstream
osion. Although SRB's are still present in relatively high numbers, we
hing of the downstream pipes which dislodges some of the biofilm

.243
ss may be more chemically controlled compared to recirculating pipes.
83.08 mg/L
dicating SRB's may be active.
32.546 mg/L
g pipes. Corrosions may be a combination of bio and chemical processes.
al: 4,495 isolates
/r/t nutrient culturing, probaby still play a role in the corrosion.
: 36 bands
m pipes.
597 cell#/mm2
d to recirculating pipes, indication that maybe process is chemically
number caused by periodic flushing of the recirculating pipes.
Bacteria the primary agent of corrosion in the recirculating pipes.
population far exceeds all others.

  - [Water Chemistry check point 3]pH: 6.378
    Reason: Neutral pH, indicating process is probably not a chemical one
  - [Water Chemistry check point 4]pH: 6.262
    Reason: Would expect more acidic pH if corrosion problem was a purely chemical process.
  - [Water Chemistry check point 3]SO4: 79.017 mg/L
    Reason: High sulfate concentration indicative of sulfate reducing bacteria. SRB are known to cause corrosion in steel piping.

Make Comment

claim > recirculating pipes > evidence

[Water Chemistry check point 3]pH: 6.378
Reason: Neutral pH, indicating process is probably not a chemical one

Comment

```
There are other possibilities for chemical
corrosion at neutral pH's- should acknowledge
this.
```

Figure 11. The critiquing interface.

Open-Ended Interface Elements for Collecting Authoring Materials

In the incremental authoring model, teachers need to find out what students need in the learning environment so that new materials can be added. Besides the student-teacher communication channel mentioned above, we use opened-ended interface elements in INDIE to collect students' inputs. Student inputs can reveal materials and feedback that students expect in the system. Inputs that do not have feedback returned indicate missing contents and provide directions for authoring.

For example, in INDIE we deliberately use a text box for students to select tests. When students enter a test name that matches one of the keywords of a test, the test will be selected. (Every test has a list of keywords because the same test can be called in different ways. For example, the culturing test can be called the gram stem test.) We do not use a menu because the text box approach lets students brainstorm what tests they need. Test names entered by the students are reviewed by the instructor and reasonable ones can be added into the learning environment. Test names can also tell the instructor what tests are needed and should be added into the system. We describe empirical results of this approach in the evaluation section.

The above discussed how we designed INDIE to make incremental authoring feasible for teachers. We believe that these design choices are essential in providing an infrastructure and tool support for incremental authoring. In the following, we describe our experience of using INDIE and the incremental authoring model to develop Corrosion Investigator.

DEVELOPMENT PROCESS OF CORROSION INVESTIGATOR

Corrosion Investigator was developed in a project funded by NSF to improve bioengineering education. We started by working with four faculty members (one from environmental engineering, one from biomedical engineering, one from learning sciences, and one from computer science and education) to design the Corrosion Investigator scenario. We chose the concepts and skills that students needed to learn, generated the corrosion challenge, explored background resources about paper processing plants, created a pipe layout for the processing plant, established the (hidden) causes of corrosion, and collected lab tests that are commonly needed to diagnose corrosion problems. The Corrosion Investigator scenario was mainly based on a faculty member's real experience in industry. The initial scenario development took about 2 months.

After the initial Corrosion Investigator scenario was developed, it was used as a course project in an environmental engineering course. At the beginning of the project, students were given a challenge statement as shown in Figure 12. The instructor acted as the liaison between the students, the paper processing company, and any commercial lab. Students asked the liaison for background information about the company and ordered lab tests. The liaison supplied information on demand, including fairly complex test results. At the end, the students submitted reports explaining their diagnoses. All communications were done via e-mail except for bi-weekly presentations where the students reported their progress.

While the Corrosion Investigator scenario was authentic and challenging, it was very labor intensive for the instructor to generate test data in response to every test request from students. Test results needed to be generated repeatedly and had to be different each time to resemble data from real labs. Furthermore, the results needed to correctly indicate the underlying cause of the corrosion problems. For students, it was time consuming to pursue the project because they needed to wait for several days for the teacher to generate test data for them.

To solve the above problems, we decided to develop the computer-based Corrosion Investigator learning environment. We also wanted to develop a software tool that allows us to easily develop learning environments similar to Corrosion Investigator.

Your team has been hired by Patriot Chemical Co. to investigate a problem they are having with their water distribution piping in their paper processing division. They historically have had severe corrosion problems associated with pipes in this system. They traditionally replace the piping—which results in severe financial loss while the system is down. Patriot is hiring your team to discover the source of the problem and provide a feasible solution that will avoid future need to replace piping. The goal of this exercise is to determine the nature of the problem and to come up with a solution as quickly and cost-effectively as possible. The plant foreman has recently noticed rust in the effluent of the system. This is usually an early indication that the pipes are beginning to fail.

We have designed this exercise to simulate as close to a "real life" scenario as possible.

Initially there will be very little information for you to work with. Using your creativity and knowledge of microbiology/engineering your team can develop a trouble shooting flow chart and dissect the problem. There is more than one way to go about getting the right answer. I will act as your liaison between Patriot and any commercial labs/services you will require to generate information crucial to solving this problem. Depending upon the information/tests you solicit, the response time will vary in accordance with the nature of the information requested. Any costs associated with requested lab tests/ information will be given as estimates to your group prior to your requesting it.

Part I) What could be the cause the problem?

Part II) How would you propose to fix it?

Background: As you investigate this challenge you need to consider multiple factors. First, the company has indicated they would like an accurate as well as cost-effective solution. In addition, they require a thorough justification of your recommendation. This requires you to draw on the knowledge presented in this class as well as information you obtain through research, data collection, consultations, etc.

Figure 12. The challenge statement in Corrosion Investigator.

We started by analyzing over 70 e-mails sent between the students and the instructor to understand what learning content should be put in the learning environment. The e-mails suggested the background information students needed, the tests they wanted to run, and the mistakes they usually made. For example, a list of questions that students asked and their corresponding answers were used as the background information in the learning environment. Five major tests that students requested became the tests in the learning environment. Additional materials and features were added in to the learning environment to make learning authentic. Random test result generation mechanism was developed to make test results realistic. The simulated cost and time mechanism were implemented to make students aware of real-life constraints.

We further designed the INDIE framework and authoring tool based on the actions and content in Corrosion Investigator. Design choices discussed in the above section were specially made to support the incremental authoring model and make sure that the infrastructure and underlying data structures were general enough so that similar learning environments can be built without changing the framework.

EVALUATION RESULTS

We conducted two small-scale studies to test the feasibility of the incremental authoring model. Both studies were conducted with students and professors at Northwestern University in 2005.

The focus of the first study was to test whether the learning environment software could sufficiently support student learning and incremental authoring. We conducted the study in the environment engineering class where the instructor was the professor who developed the initial Corrosion Investigator scenario and had run the scenario without the software in his previous class. We asked the instructor to teach his class exactly the same way as he did before except using the software for the Corrosion Investigator challenge. This allowed us to compare student learning outcomes between the two classes to evaluate the impact of the software. There were six first-year graduate students in the class. They worked in two groups of three on the Corrosion Investigator challenge as their final course project. During the project, students worked with the software learning environment to run tests, collect data, and construct reports. They contacted characters in the challenge to request additional information. The instructor role-played the characters and answered students' requests. He provided coaching through the critiquing interface in the learning environment. Furthermore, the instructor identified missing test names and options by reviewing student activities and added them into the learning environment using the authoring tool. The project lasted 3 weeks. After the completion of the project, we gave the students and the instructor a survey regarding their experience with the software. We recorded what the instructor authored during the study.

After the first study, we conducted the second study. The focus of the study was to investigate whether instructors other than the original author of the scenario could deliver the challenge and extend the learning content. Five second-year graduate students and two professors (other than the one in the first study) in the environmental engineering department volunteered to participate in our study. Students were given 3 weeks to complete the challenge. They worked in two groups. The two professors facilitated the project as the instructor did in the first study by providing feedback to the students whenever needed. After 3 weeks, students completed the challenge. We recorded what the two professors' authored during the study. We will present the authoring data in the following section (together with the ones collected from the first study).

Evaluation results presented below are based on two studies with a total number of 11 students and three professors. They are suggestive but by no means proven because of the small sample size. The goals of our studies are to test whether the incremental authoring model is feasible and to show how teachers can use it. While the data that we obtained are limited, they do provide an example of how INDIE can facilitate students learning and incremental authoring. Future work will be conducted to verify the findings in the current studies with larger numbers of students and teachers, and assess the generalizability and limitation of the software and authoring model.

In the following, we present results regarding students' evaluation, instructors' evaluation, and incremental authoring of tests and critiques.

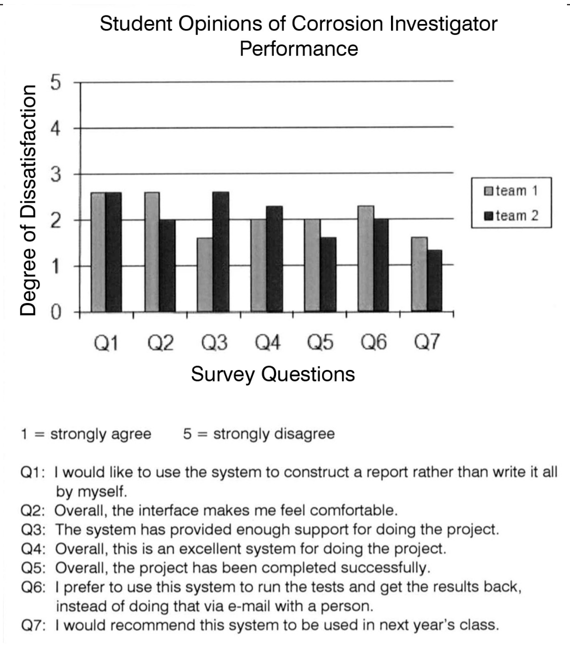
STUDENTS' EVALUATION

In our first study, we obtained students' evaluation of the Corrosion Investigator learning environment through a survey. The survey asked students to rate the performance of the learning environment in supporting their project.

Figure 13 shows the results from the survey. Overall, students agreed that the learning environment was satisfying in delivering the challenge. They agreed that the system provided enough support for them to successfully complete the project. They would recommend the system to be used by other students.

Students largely benefited from the immediate feedback generation from the system. The project time was reduced from 8 weeks (the time it took when the challenge was delivered without the software and the instructor needed to generate the test results manually) to 3 weeks. This was evident in students' answers to the question "What did you like the most about the system?":

> results are immediate. you can plan around time spent on each test.
>
> automatically run tests and report time and cost.
>
> the instant feedback on tests results. did not have to wait for a person to email results back to me.
>
> fast response time.



1 = strongly agree 5 = strongly disagree

Q1: I would like to use the system to construct a report rather than write it all by myself.
Q2: Overall, the interface makes me feel comfortable.
Q3: The system has provided enough support for doing the project.
Q4: Overall, this is an excellent system for doing the project.
Q5: Overall, the project has been completed successfully.
Q6: I prefer to use this system to run the tests and get the results back, instead of doing that via e-mail with a person.
Q7: I would recommend this system to be used in next year's class.

1 = strongly agree 5 = strongly disagree

Q1: I would like to use the system to construct a report rather than write it all by myself.
Q2: Overall, the interface makes me feel comfortable.
Q3: The system has provided enough support for doing the project.
Q4: Overall, this is an excellent system for doing the project.
Q5: Overall, the project has been completed successfully.
Q6: I prefer to use this system to run the tests and get the results back, instead of doing that via e-mail with a person.
Q7: I would recommend this system to be used in next year's class.

Figure 13. Student evaluation of Corrosion Investigator performance.

The design goal of the learning environment was to facilitate students' problem-solving by providing them with necessary information and tools to reduce their project time. While there are still usability issues in the learning environment (e.g., the interface for retrieving tests was not easy to use), the response from the students suggest that the learning environment has met the initial design goal. In the following, we further show that the instructor considered students' learning outcome remains the same when compared to the one gained in the class when the software was not used.

Instructor's Evaluation

In our first study, we obtained instructor's evaluation of the learning environment. We gave the instructor a survey asking him to compare his experience in delivering the scenario using the software with his experience in delivering the scenario without the software.

Figure 14 shows the responses from the instructor in the survey. The instructor considered the effort in delivering the scenario and the time students needed to complete the scenario were significantly less when using the software. The quality of the data generated by the system was slightly better. For the instructor, the use of the software reduced his workload from 24 total man-hours to 4. It was most evident in reducing the work in generating test results. When asked "What did you like the most of the system?" the instructor reported:

> It was fairly self-sufficient, gave quick response, easy to view history of system (what operations were ordered, etc).

Furthermore, the instructor considered the quality of students' work and their learning of the target skills remained the same when compared to the ones in the class where the challenge was delivered without the software.

The above data show that Corrosion Investigator greatly helped the instructor deliver the problem-based learning module while maintaining the student learning quality. This is consistent with our initial design goal.

Question	Answer (much less) 1 2 3 4 5 6 7 (much more)
The effort involved in delivering the scenario with the software	1
The time students needed to complete the scenario using the software	1
Question	Answer (much better) 1 2 3 4 5 6 7 (much worse)
The quality of the simulated data given to the students by the software	3
The quality of the students' final reports after using the software	4
The student learning of the target skills after using the software	4

Figure 14. Instructor's evaluation of Corrosion Investigator.

Evaluation Results for Incremental Authoring

During both of our studies, we recorded the learning content authored by the teachers. In the following, we present results obtained from the two studies regarding test authoring and critique authoring.

Test Authoring

Corrosion investigator initially contained 39 test names to match student inputs. During the two studies, teachers added 34 more test names into Corrosion Investigator. This resulted in an increase of 87% of test names in the system. Figure 15 shows the increase of test names for each test. (One test can have multiple names.) Furthermore, in the second study, one professor found that "dissolved Fe" should be added as a test option for the water chemistry after he received students' request for the test option. Figure 16 shows the test data added into the system for the test option.

The addition of the large number of test names and an important test option suggested that even with careful preparation, it is still difficult to make sure that the learning environment contains all the necessary content. It is necessary to allow teachers to add new content into the system during runtime.

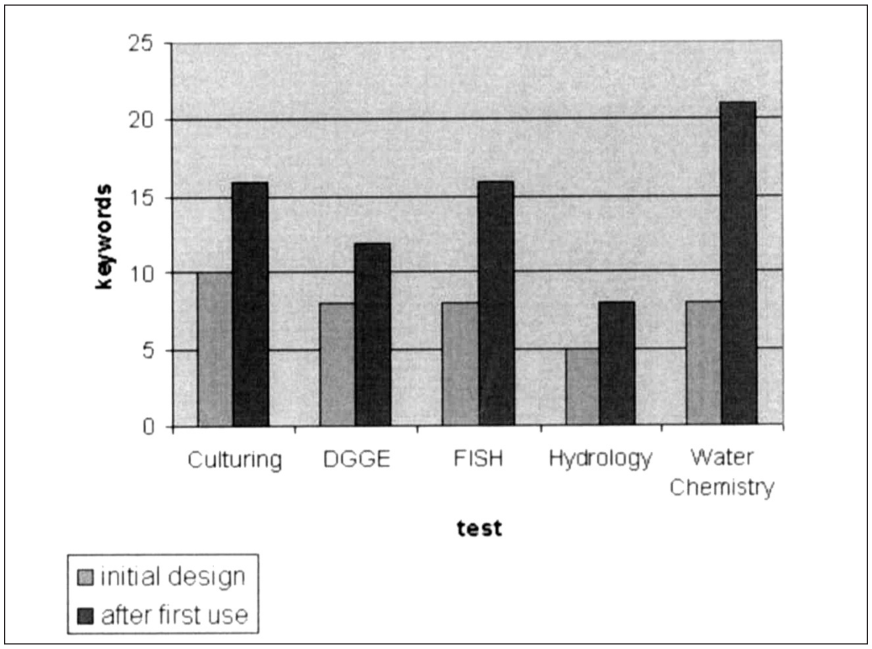


Figure 15. Keywords for the tests in Corrosion Investigator.

Location of Sample	Results for "dissolved Fe" (mg/L)
Checkpoint 1 Checkpoint11,Checkpoint12	0.01 ~ 2
Checkpoint 2, Checkpoint 3, Checkpoint 4, Checkpoint 5, Checkpoint 6, Checkpoint 7, Checkpoint 8	1 ~ 5
Checkpoint 9, Checkpoint 10	0.01 ~ 2

Figure 16. Test results for "dissolved Fe" in the water chemistry test.

We further conducted a survey with the instructor in the first study regarding his use of the authoring tool. Results show that it was not hard for the instructor to understand the data structure of the learning content in the system and map his knowledge to the data structure (see Figure 17). When asked "what did you like least about the system? Would you have preferred more facilities in the system?" The teacher responded:

> (the interface is) not very intuitive. need some way of guiding user through process.

We believe the above problem is caused by the lack of directions provided in the authoring interface. Our future work would be to add a wizard or more instructions on the authoring interface so that authors can be guided through the authoring process.

When responding to "what you like the most about this system," the instructor reported:

> Once I knew what I was doing, it was easy to manipulate system.

We believe this is caused by the consistent look-and-feel of the authoring interface. After the initial learning stage, the instructor could easily use the interface for authoring.

The survey also includes questions regarding the usefulness of specific features in the authoring tool. Results show that these features were all considered very helpful (see Figure 18).

The above results show that teachers are capable to use the INDIE authoring tool to add missing content into the learning environment during runtime.

Critique Authoring

Critique authoring has always been a difficult part of learning environment authoring because it is difficult to know what mistakes students commonly make

Question	Answer (very easy) 1 2 3 4 5 (extremely hard)
How difficult is it for you to understand the data structure in the system?	3
How difficult is it for you to transform your knowledge to fit into the data structure in the system?	1
How difficult is it for you to enter the specification for a test?	4
How difficult is it for you to enter test results?	4
How difficult is it for you to interact with the system?	3

Figure 17. Survey results regarding the instructor's authoring experience.

Question	Answer (strongly agree) 1 2 3 4 5 (strongly disagree)
It is very helpful to have an automatic checker to verify the consistency of the knowledge in the system.	1
It is very helpful to have a complete view of the domain and scenario knowledge in the system.	1
It is very helpful to check the student view of a test.	1

Figure 18. Instructor's evaluation of the INDIE authoring tool.

and what critiques to give for those mistakes. Our goal is to use incremental authoring to first collect common critiques, and then based on theses critiques, design an interface to help teachers apply or automate these critiques. The following describe the results regarding the first step, which is to use incremental authoring to collect critiques. During the two studies, a total number of 32 critiques were recorded in the system. Appendix A shows some examples of these critiques. We analyzed all the critiques and found that they could be categorized into three types.

The first type confirms the correctness of the student's work. For example,

> That is correct- H2S a byproduct of SRB metabolism.

The second type points out that the student work is wrong. For example,

> This is NOT evidence supporting chemical corrosion as a cause.

The third type asks for more data or explanation. It can directly ask for explanations, for example,

> More detail could be supplied in regards to the nature of the corrosion.
>
> There are other possibilities for chemical corrosion at neutral pH's—should acknowledge this.

or can provide directions for further investigation, for example,

> Why is corrosion the worst here instead of other areas of piping between the primary treatment plant and the recirculating pipes?

or it can present data that challenge students' understanding, for example,

> SRB counts are very low compared to the other corrosion site.

The above critiques provided the basis for us to design the automatic critiquing mechanism. Our next step is to develop tools for teachers to reuse these critiques. We plan to use Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997; Landauer, Foltz, & Laham, 1998) to compare students' work against stored examples of mistakes and suggest corresponding critiques. Teachers will be able to review automatically generated critiques and send them to students. We have successfully experimented incremental critiquing authoring in another system called Java Critiquer (Qiu & Riesbeck, 2008). The Java Critiquer provides an environment where teachers can critique students' Java code and incrementally author critiques in the system for reuse. We aim to use similar techniques to allow the system to automatically suggest critiques and allow the teachers to review them.

DISCUSSION AND FUTURE WORK

The target audiences for INDIE learning environments are students in colleges. In our studies, first and second year graduate students successfully accomplished the main tasks of an INDIE-authored learning environment. K-12 students may have difficulties in using such learning environments because compared to students in our studies, they may not have strong interests in the subject area and could easily lose interest in using the text-based learning environment. Interactive learning environments built for K-12 students (e.g., Lajoie et al., 2001; Liu et al., 2002) often use rich media such as videos and graphics to attract students' attention. However, INDIE learning environments are largely text-based because we focus on the logical diagnostic reasoning aspects of the challenges and want to keep authoring simple. While multimedia content can be embedded into Webpages and uploaded into INDIE learning environments, creating them on the fly would be difficult for teachers. Therefore, the Web-based INDIE learning environments do not provide immersive multimedia to maintain students' interests. Furthermore, K-12 students may not have strong self-learning capability as students in our studies. They often need more guidance from their teachers outside of the software learning environment. This would make the authoring of the learning content difficult because the incremental authoring model relies on having students and teachers communicate through the learning environment so that teachers can save the feedback as new learning content. Given the above issues, we believe that the incremental authoring model is most appropriate to create learning environments for students in high school and college.

To use the incremental authoring model to create learning environments, teachers need to have a deep understanding of the problem-based learning pedagogy and know how to create a problem-based learning module. While many teachers may be interested in delivering problem-based learning, not all of them know how to develop a challenging scenario with proper learning activities embedded. One way to use the incremental authoring model is to have domain experts and learning scientists create a number of initial modules and have teachers extend them during teaching. The initial modules will include the challenges that students need to solve and the major tasks that students need to perform. A teacher can pick the module that he or she wants to use and extend the module according to students' needs. We believe that because the module is initially created by subject-matter experts based on sound educational principles, teachers will be less likely to fail to follow the problem-based learning pedagogy. While this approach does not guarantee the quality of the learning content, we believe that it is worthwhile to provide teachers the capability to create their own learning environments.

INDIE learning environments are best used in situations where teachers actively monitor and guide student learning. We have tested the INDIE learning environment in a course project and a volunteer project. In both projects, teachers paid close attention to students' progress and provided them with coaching and critiquing. This is critical because the learning environment in the incremental authoring model does not have all the necessary feedback for students. Teachers need to identify missing feedback and provide it to students. We aim to create a teacher-computer collaboration system where students can receive feedback from both the computer and the teacher.

We believe the incremental authoring model could be extended to support the development of other knowledge-based educational systems, especially text-based educational systems, besides interactive learning environments. The key to the success of the incremental authoring model is to establish a feedback loop where the teacher needs to provide feedback to students through the system. We have experimented the incremental authoring model in a critiquing system for software programming (Qiu & Riesbeck, 2008). The system allows teachers to critique students' programming code and add new critiques into the system. The teacher can also refine existing critiques and create patterns to automate critiques. We tested the system with two instructors using the system in university-level introductory programming courses. After one-year's use, a total number of 232 critiques were collected and remained relatively stable for over a year. In our future work, we aim to experiment the incremental authoring model with other types of learning systems in different educational settings to further evaluate its feasibility and effectiveness.

CONCLUSION

In this article, we described an incremental authoring model for developing interactive learning environments. This model includes an instructor in the feedback loop to complement system feedback and incrementally author the learning content in response to demands from actual students. We described INDIE, a learning environment authoring toolkit, as an example to show how to support the incremental authoring model. INDIE uses a Web-based client-server architecture to allow teachers to author the learning environment anytime anywhere. It provides a domain-independent interface framework with an authoring tool for teachers to perform authoring without programming. It uses a hybrid feedback mechanism to allow teachers to receive requests from students and complement system feedback. It uses open-ended interface elements to help teachers collect students' inputs and identify materials for authoring. These design choices in INDIE enable the key components in the incremental authoring model to ensure that teachers can perform authoring at runtime.

We presented our development experience of Corrosion Investigator, a learning environment delivered by INDIE, to show how the incremental authoring model can be implemented. Corrosion Investigator has been used twice, once in a class and once in a volunteer project. Feedback from the students and instructor suggests that Corrosion Investigator successfully facilitated the delivery of a problem-based learning module. Results regarding test and critique authoring show that INDIE is capable to support incremental authoring during runtime.

The incremental authoring model explores a vision of developing knowledge-based educational systems with less upfront development effort. It avoids the need to anticipate and implement all possible situations upfront and allows the system to gradually evolve into a complete system after deployment. With the widespread use of Web technology, we believe the model is feasible and promising in facilitating the authoring and customization of knowledge-based educational systems.

APPENDIX A
Critiques entered by instructors in Corrosion Investigator

Student work	Critique
Claim: Acidic pH and chemical oxidation of pipes the main cause of corrosion in the downstream pipes. pH level is acidic enough to cause corrosion. Although SRB's are still present in relatively high numbers, we feel that they are preset due to periodic flushing of the downstream pipes which dislodges some of the biofilm population.	This captures the essence of the problem occurring in the downstream pipes. More detail could be supplied in regards to the nature of the corrosion. What type of corrosion could be occurring? Why is corrosion the worst here instead of other areas of piping between the primary treatment plant and the recirculating pipes?
Test Result: [Water Chemistry check point 9]SO4: 83.08 mg/L Reason: High sulfate is still present, indicating SRB's may be active.	This is NOT evidence supporting chemical corrosion as a cause.
Test Result: [Water Chemistry check point 3] pH: 6.378 Reason: Neutral pH, indicating process is probably not a chemical one.	There are other possibilities for chemical corrosion at neutral pH's—should acknowledge this.
Test Result: [Water Chemistry check point 3]H2S: 42.204 mg/L Reason: Rotten egg-like odor indicative of sulfate reduction. High H2S concentration is indicative of SRB populations	That is correct—H2S a byproduct of SRB metabolism.
Test Result: [Water Chemistry check point 9]H2S: 32.546 mg/L Reason: Not as high as in recirculating pipes. Corrosions may be a combination of bio and chemical processes.	Not well explained—is H2S derived from activity at that location, or is it left over from water derived from flushed recirculating water.
Test Result: [Culturing check point 9, SRB,25C] Total: 4,495 isolates Reason: SRB have the most isolates w/r/t nutrient culturing, probably still play a role in the corrosion.	SRB counts are very low compared to the other corrosion site.

APPENDIX A (Cont'd.)

Student work	Critique
Test Result: [Water Chemistry check point 3]SO4: 79.017 mg/L Reason: High sulfate concentration indicative of sulfate reducing bacteria. SRB are known to cause corrosion in steel piping.	This is wrong—High SO4 concentration indicates that there are substrates present for SRB growth.
Test Result: [Water Chemistry check point 4] temperature: 29.796°C Reason: Temperature is ideal for SRB growth.	Why is it ideal? What is the temperature optimal for most SRB's? This is not known . . .
Test Result: [Water Chemistry check point 3]pH: 6.554 Reason: The pH at point 3 is neutral.	While pH of 6.6 is near neutral, this does not by itself support that SRB are active. SRB activity would produce base and may increase the pH. Do you have a pH comparison?
Test Result: [Water Chemistry check point 10]pH: 3.946 Reason: pH is low, indicating that there is acid production in the downstream pipes.	Yes, a low pH implies acid production, which is part of sulfide oxidation. From where does the oxygen come?
Test Result: [Water Chemistry check point 9]SO4: 77.228 mg/L Reason: SO4 concentration is lower comparing to the upstream.	This is lower than check point 3, and it implies that sulfate was reduced after checkpoint 3. This may be evidence of sulfate reduction in the recirculating loop, not evidence for what is happening in the downstream pipes.
Test Result: [Water Chemistry check point 10]DO: 6.889 mg/L Reason: DO is high.	A high D.O. supports that O2 can be the electron acceptor for corrosion, but it is counter to having sulfide oxidized to SO42-, which is strongly oxygen consuming. So, from where does the oxygen come to allow sulfide oxidation?

APPENDIX A (Cont'd.)

Student work	Critique
Claim: RECIRCULATING PIPES: According to our test results, the excessive corrosion in both the recirculating and downstream pipes is likely due to the presence of iron-oxidizing bacteria. The livelihood of these microbes, which use hydrogen sulfide as an electron donor, is probably supported by anaerobic sulfate-reducing bacteria. In spite of the ample amounts of oxygen throughout the system. SRB's can create anoxic conditions within a deep biofilm, where they can take advantage of the abundance of sulfate provided by the Biodex sticking agent. It is our recommendation that a new sticking agent that does not contain sulfate, such as talc, be used instead of Biodex.	You are suggesting that iron-reducing bacteria is present at two checkpoints (i.e., circulating and downstream pipes). Isn't the pH of the water sample too high to maintain iron-reducing bacteria in both check points? What is the optimum pH range for iron-reducing bacteria? You also acknowledge that the dissolved oxygen level is high especially in checkpoint 9. What would this mean? What is/are the substrate(s) for iron-reducing bacteria? Are you eliminating the possibility of SRBs? why or why not? If the sticking agent does not contain sulfate, how would this help get rid of iron-reducing bacteria? Any thoughts?
Test Request: DGGE Scheduled Date: 25 Test Parameters: Location of Sample: check point 5; Primer Set: sulfate reducers; Reason: Sulfate reducers may be the producers of H2S in recirculating pipes	This is a good point. How can you make sure of this, any cheap way of finding this out?
Test Request: Water Chemistry Scheduled Date: 28 Test Parameters: Test variable: Cl-; H2S; SO4; dissolved oxygen; pH; Location of Sample: check point 9; Reason: Compare S concentrations between recirculating and downstream pipes	What is your conclusion based on this piece of information? Would temperature suggest anything? How about DO and pH? Would they suggest anything?

REFERENCES

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)* (pp. 61-70). Berlin: Springer Verlag.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Barrows, H. S. (2000). *Problem-based learning applied to medical education*. Springfield, IL: Southern Illinois University Press.

Beck, K. (2000) *Extreme programming explained—Embrace change.* White Plains, NY: Addison-Wesley.

Bell, B. L., Bareiss, R., & Beckwith., R. (1994). Sickle cell counselor: A prototype goal-based scenario for instruction in a museum environment. *Journal of the Learning Sciences, 3,* 347-386.

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

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *Journal of the Learning Sciences, 2,* 141-178.

Cockburn, A. (2002) *Agile software development.* White Plains, NY: Addison-Wesley.

Collins, A., Brown, J. S., & Newman, S. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: theoretical and methodological issues. *Journal of the Learning Sciences, 13,* 15-42.

de Jong, T., & W. R. van Joolingen. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research, 68,* 179-201.

Derry, S. J., Hmelo-Silver, C. E., Nagarajan, A., Chernobilsky, E., & Beitzel, B. (2006). Cognitive transfer revisited: Can we exploit new media to solve old problems on a large scale? *Journal of Educational Computing Research, 35,* 145-162.

Dobson, W. D. (1998). *Authoring tools for investigate and decide learning environments.* Unpublished manuscript, Northwestern University.

Dochy, F., Segers, M., Van den Bossche, P., & Gijbels, D. (2003). Effects of problem-based learning: A meta-analysis. *Learning and Instruction, 13,* 533-568.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *Journal of the Learning Sciences, 11,* 105-121.

Fischer, G. (1998). Seeding, evolutionary growth and reseeding: Constructing, capturing and evolving knowledge in domain-oriented design environments. *International Journal of Automated Software Engineering, 5,* 447-464.

Fischer, G., & Ostwald, J. (2002). Seeding, evolutionary growth, and reseeding: Enriching participatory design with informed participation. In T. Binder, J. Gregory, & I. Wagner (Eds.), *Proceedings of the Participatory Design Conference* (PDC'02) (pp. 135-143). California: Malmö University.

Hmelo-Silver, C. E., & Barrows, H. S. (2006). Goals and strategies of a problem-based learning facilitator. *Interdisciplinary Journal of Problem-Based Learning, 1,* 21-39.

Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and Clark (2006). *Educational Psychologist, 42,* 99-107.

Hoffman, B., & Ritchie, D. (1997). Using multimedia to overcome the problems with problem based learning. *Instructional Science, 25,* 97-115.

Hsieh, P., Halff, H., & Redfield, C. (1999). Four easy pieces: Developing systems for knowledge-based generative instruction. *International Journal of Artificial Intelligence in Education, 10,* 1-45.

Koedinger, K., Aleven, V., Heffernan, N., McLaren, B., & Hockenberry, M. (2004). Opening the door to nonprogrammers: Authoring intelligent tutor behavior by demonstration. In *Proceedings ITS-2004* (pp. 162-174). Berlin: Springer.

Koh, G. C., Khoo, H. E., Wong, M. L., & Koh, D. (2008). The effects of problem-based learning during medical school on physician competency: A systematic review. *Canadian Medical Association Journal (CMAJ), 178,* 34-41.

Lajoie, S. P., Lavigne, N. C., Guerrera, C., & Munsie., S. (2001). Constructing knowledge in the context of BioWorld. *Instructional Science, 29,* 155-186.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review, 104,* 211-240.

Landauer, T. K., Foltz, P. W., & Laham., D. (1998). An introduction to latent semantic analysis. *Discourse Processes, 25,* 259-284.

Larman, C. (2003). *Agile and iterative development: A manager's guide*. White Plains, NY: Addison-Wesley Professional.

Liu, M, Williams, D., & Pedersen, S. (2002). Alien rescue: A problem-based hypermedia learning environment for middle school science. *Journal of Educational Technology Systems, 30,* 255-270.

Mergendoller, J. R., Maxwell, N. L., & Bellisimo, Y. (2006). The effectiveness of problem-based instruction: A comparative study of instructional method and student characteristics. *Interdisciplinary Journal of Problem-Based Learning, 1,* 49-69.

Molenda, M., Pershing, J. A., & Reigeluth, C. M. (1996). Designing instructional systems. In R. L. Craig (Ed.), *The ASTD training and development handbook* (4th ed., pp. 266-293). New York: McGraw-Hill.

Munro, A., Johnson, M. C., Pizzini, Q. A., Surmon, D. S., Towne, D. M., & Wogulis, J. L. (1997). Authoring simulation-centered tutors with RIDES. *International Journal of Artificial Intelligence in Education, 8*(3-4), 284-316.

Murray, T. (2003). An overview of intelligent tutoring system authoring tools: Updated analysis of the State of the art. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring tools for advanced technology learning environments* (pp. 491-545). Amsterdam: Kluwer Academic Publishers.

Murray, T., Blessing, S., & Ainsworth S. E. (2003). *Authoring tools for advanced technology learning environments.* Amsterdam: Kluwer Academic Publishers.

Norman, D. A., & Draper, S. W. (Eds.). (1986). *User-centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Pawson, R., & Mathews, R. (2002). *Naked objects.* Chichester, England: Wiley.

Qiu, L. (2005). *A web-based architecture and incremental authoring model for interactive learning environments for diagnostic reasoning*. Unpublished manuscript, Northwestern University.

Qiu, L., & Riesbeck, C. K. (2008). An incremental model for developing educational critiquing systems: Experiences with the Java Critiquer. *Journal of Interactive Learning Research, 19.*

Reeves, B., & Nass, C. (1996). *The media equation*. Cambridge: SLI Publications, Cambridge University Press.

Roschelle, J., Kaput, J., Stroup, W., & Kahn, T. (1998). Scalable integration of educational software: Exploring the promise of component architectures. *Journal of Interactive Media in Education, 98*(6), 1-31.

Royce, W. (1970, August). Managing the development of large software systems. *Proceedings of IEEE WESCON, 26*, 1-9

Schank, R. C. (1982). *Dynamic memory: A theory of reminding and learning in computers and people.* Cambridge: Cambridge University Press.

Schank, R., Fano, A., Bell, B., & Jona, M. (1993). The design of goal-based scenarios. *Journal of the Learning Sciences, 3,* 305-345.

Schuler, D., & Namioka, A. (1993). *Participatory design: Principles and practices.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Sparks, R., Dooley, S., Meiskey, L., & Blumenthal, R. (1998). The LEAP authoring tool: Supporting complex courseware authoring through reuse, rapid prototyping, and interactive visualizations. *International Journal of Artificial Intelligence in Education, 10,* 75-97.

Torp, L., & Sage, S. (2000). *Problems as possibilities: Problem-based learning for K-16 Education.* (2nd ed.). Alexandria, VA: ASCD.

van Joolingen, & de Jong, T. (1996). Design and implementation of simulation-based discovery environments: The SMISLE solution. *International Journal of Artificial Intelligence in Education*, *7*(3/4), 253-276.

Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research and Development, 53,* 5-23.

Wiggins, G. (1992). Creating tests worth taking. *Educational Leadership, 49,* 26-33.

Wilson, J., & Rosenberg, D. (1988). Rapid prototyping for user interface design. In M. Helander (Ed.), *Handbook of human-computer interaction* (pp. 859-875). New York: North-Holland.

Woolf, B. P, & Cunningham, P. A. (1987). Multiple knowledge sources in intelligent teaching systems. *IEEE Expert, 2,* 41-54.

Direct reprint requests to:

Dr. Lin Qiu
Department of Computer Science
State University of New York at Oswego
Oswego, NY 13126
e-mail: lqiu@oswego.edu