A Software Engine for Building Investigate and Decide Learning Environments

Lin Qiu and Christopher K. Riesbeck
Department of Computer Science
Northwestern University
1890 Maple Avenue.
Evanston, Illinois 60201 USA
{qiu, riesbeck}@cs.northwestern.edu

Abstract
We describe a new version of Indie, a software engine for building Investigate and Decide learning environments. Indie is a content-independent tool that constructs web-based GBS learning environments from XML files specifying domain content. In Indie systems, students can run simulated experiments, construct reports arguing for different claims, and submit these reports for critiquing. Indie has been used to construct Corrosion Investigator, a learning environment for engineering undergraduates in the domain of biofilms.

Keywords
Goal-based scenarios, Investigate and Decide, interactive learning environment, context-independent engine, web-based tool, authoring tool.

Introduction
An important skill engineering students need to learn is how to solve problems by running experiments and using the results to support or refute possible diagnoses. We call such activity "Investigate and Decide" [3]. In such an activity, students need to learn how to choose the right experiments to run, interpret the relevance of the results gained, and build a well-supported claim based on the evidence. A major problem with learning such skills in school is that experiments can be very expensive and time-consuming. For example, collecting water samples at a large processing plant and culturing bacteria can cost hundreds to thousands of dollars and take several weeks. Furthermore, students need coaching and feedback in order to use lab results efficiently and effectively. One way to avoid these problems is to have students work in a supportive simulated environment.

A graduate-level engineering course at Northwestern University was recently taught using a simulated Investigate and Decide framework. The course was about the biological and engineering concepts and consequences of biofilms. In the class, students took the role of consultants helping a client (a paper processing plant) diagnose and remediate recurring pipe corrosion. The instructor acted as a liaison between the students, the client, and various simulated commercial laboratories. Using email, students asked for background information about the company and ordered lab tests. The instructor generated this information on demand, including fairly complex test results. At several points in the course, students gave presentations on what they'd learned, what they thought was causing the corrosion, and what should be done. The instructor critiqued on how well the claims were supported. Students then tried to improve their work according to the critiques. All communications were done via email.

Though the task was engaging and challenging, generating plausible data for each experiment by hand was labor intensive. For realism, the instructor had to generate slightly different sets of data for each test ordered by students. As a result, students still needed to wait for several days before getting test results form the instructor, which hindered the learning process.

To solve these problems, we have built a second generation version of a software engine and authoring tool called Indie[3], for creating Investigate and Decide environments. We used Indie to build the Corrosion Investigator application, a computer-based version of the biofilm course.

System Description
Indie is a content-independent Java-based software engine. It reads in XML files that describe the domain knowledge, scenario and outputs a learning simulation. Indie engine provides a common interface, including support for a splash screen, a welcoming "challenge" document, a "background" screen where

students can browse materials describing the scenario and domain content, a "lab" interface where students can order tests and collect results, and a "report" interface where students can construct arguments for and against possible diagnoses, using the evidence gathered from the tests.

To create a specific application, such as Corrosion Investigator, an author creates HTML documents and images for the splash screen, challenge document, and background reference material, and an XML file describing the tests students can perform, the results those tests can produce in general, and will produce in the current scenario, the possible diagnoses that the student has to argue for and against, and, optionally, how the test results relate positively or negatively to the different possible diagnoses.

An Example: Corrosion Investigator

In Corrosion Investigator, the challenge page tells students that they are hired by Patriot Chemical Co. to investigate a problem the company is having with its water distribution piping in its paper processing division. The point of this is to set up the goal for students and stimulate students to actively engage in the problem as a realistic challenge. The students can then go to the Background screen to read further about the challenge, the client, and the domain. (Fig. 1).

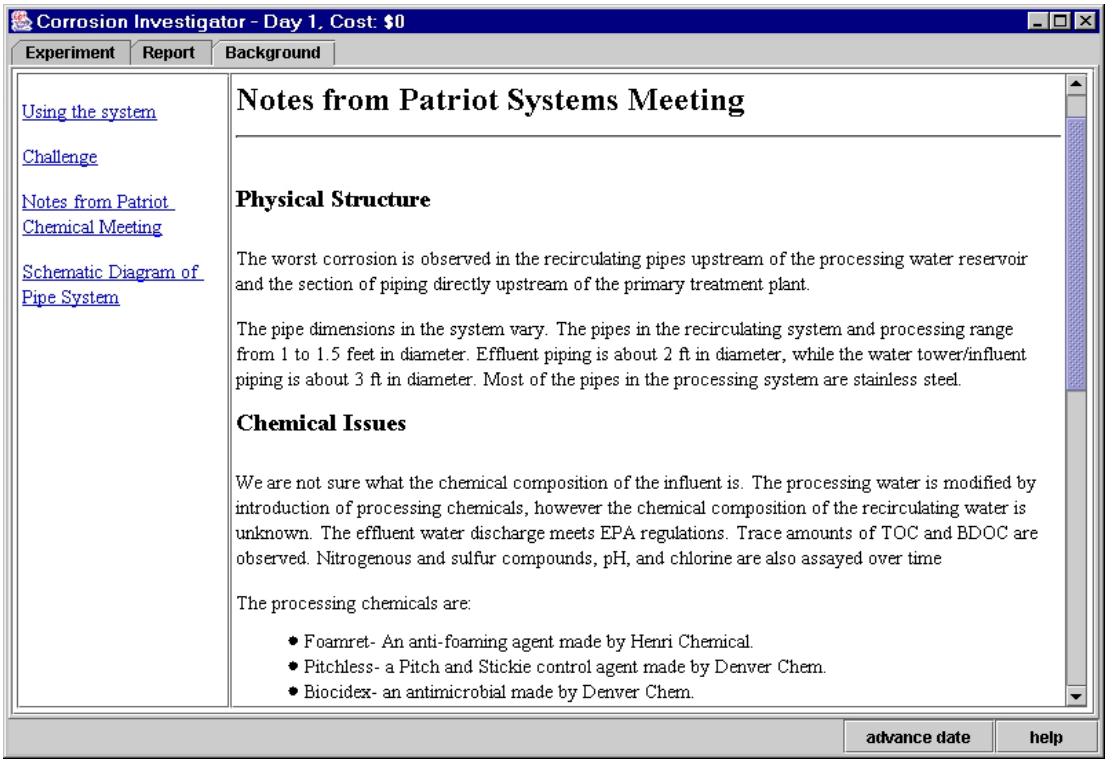

Figure 1: The Background screen in Corrosion Investigator.

To run tests, students go to the Experiment page (Fig. 2). Here, students can select tests from a pull-down list. Currently, there are five tests in Corrosion Investigator. A few more will be added. The tests are realistic and many are common, but not all are appropriate in the current scenario. Students are critiqued for running tests that are clearly irrelevant.

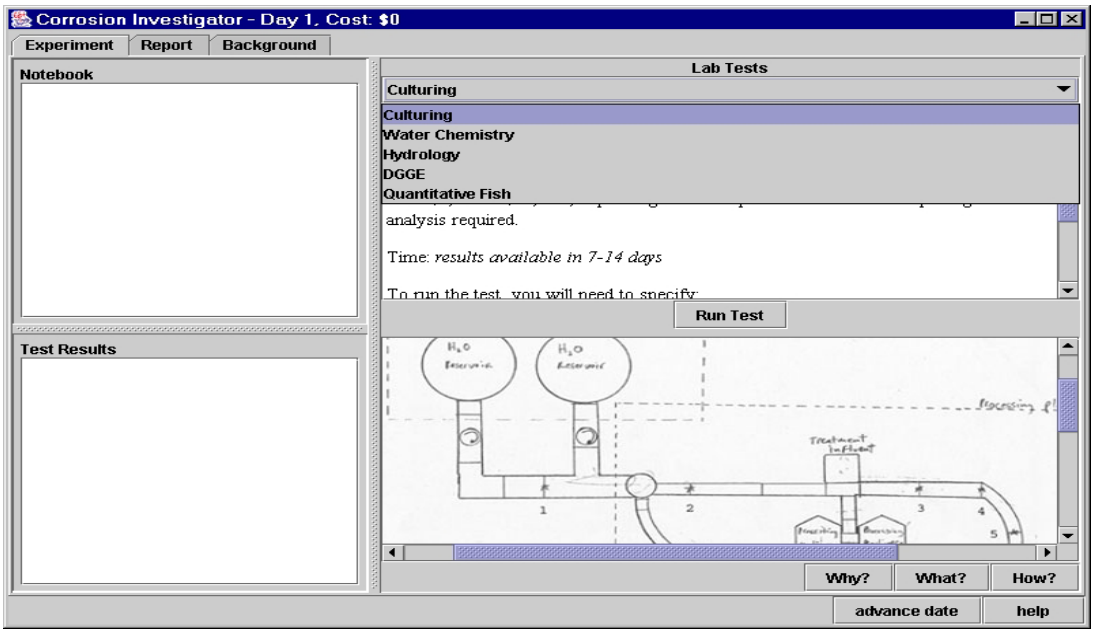

Figure 2: The Experiment screen in Corrosion Investigator.

Tests can be quite complex. For example, if students select the Culturing test, a dialog box appears where students have to specify options for the test (Fig. 3). In this case, students need to specify check points (locations where water samples should be taken), one or more culturing media, and what type of analyses to perform. The more options that are selected, the more expensive the test will be. The total cost of the test is dynamically calculated and displayed. Costs are a realistic mechanism for getting students to think hard about what tests to run. Tests also taken some number of days to run, so students have to plan what tests to do first so as not to take too long (in simulated time). Student performance will be evaluated by the instructor based on the amount of time and money they spend.

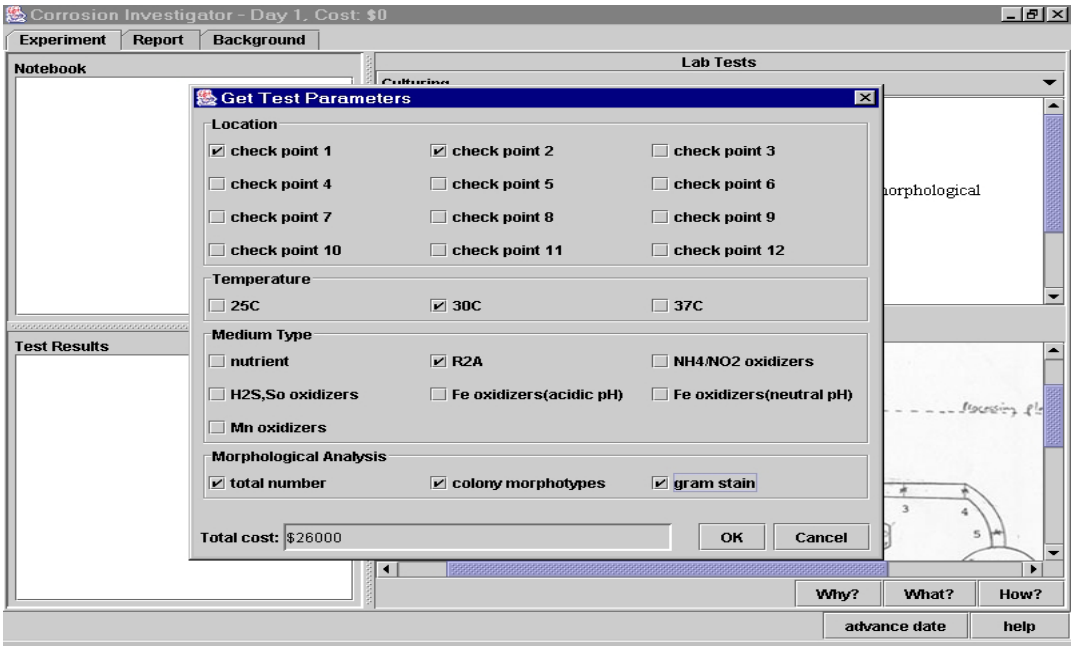

Figure 3: Parameter dialog box for Culturing Test

When the test results become available, they appear in both the Notebook and Test Results area (Fig. 4). Test Results area displays the test results in a readable manner. The Notebook records all the test results that the user has received in a list that can be used on the Report screen. Most of the numbers in the test results are generated randomly based on constraints specified in the domain files. For instance, in Figure 4, each number in the V column has to be less than the numbers in the positive and negative columns in the same row. Each number under Gram Stain column has to be the sum of the rest numbers in the same row. The total of numbers under V column has to be less than ten percent of the total amount of bacteria. Generating such complicated data sets took us a fair amount of time.

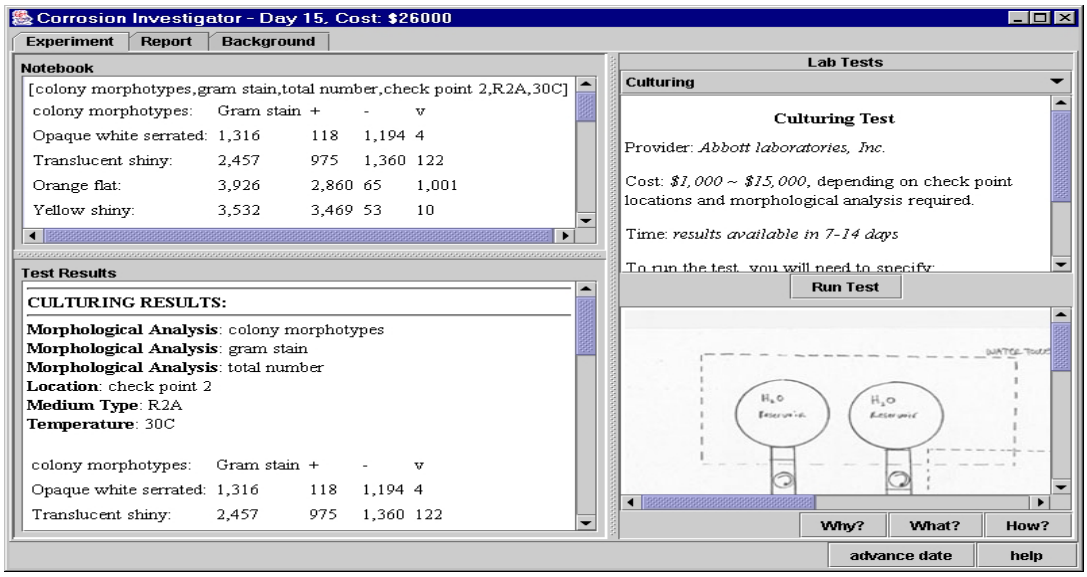

Figure 4: Test results for Culturing Test in Corrosion Investigator.

When students feel they have gathered enough information, they can go to the Report screen (Fig. 5), select one of several claims, and apply evidence in the Notebook towards that claim. In Corrosion Investigator, we currently have two broad claims: chemical corrosion and biological-induced corrosion. Students need to use many pieces of evidences to support their claim, because the strength of the argument, and its freedom from irrelevant evidence, is a key area for critiquing. After finishing constructing the argument, students click the Submit button to submit their reports.

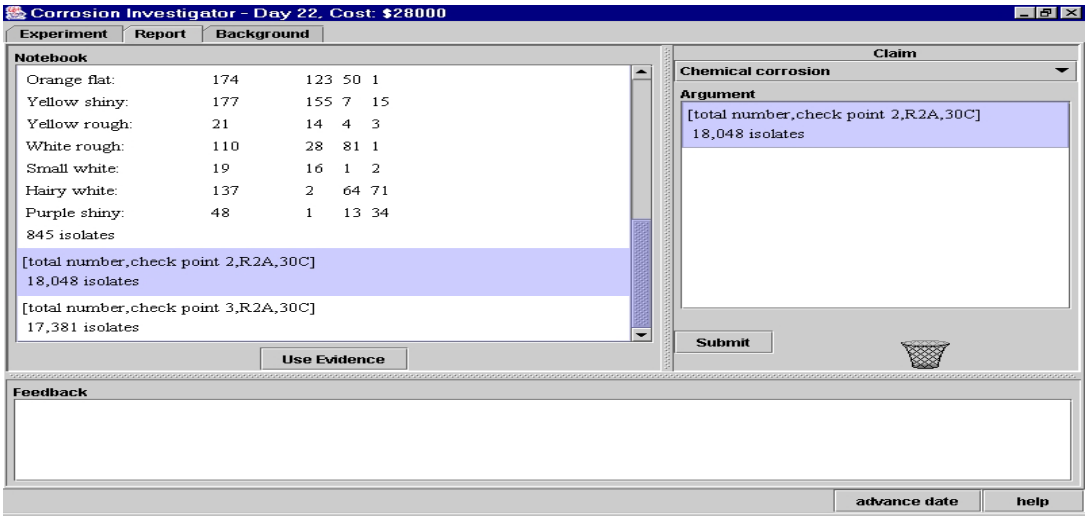

Figure 5: The Report screen in Corrosion Investigator.

After students submit a report, the system emails the report and the data in the Notebook to the instructor. The instructor reviews the report and emails feedback to the students. Students then continue working based on the feedback. If critiquing rules have been specified in the domain files, critiques from these rules are added to the report sent to the instructor.

Related Work

Investigate and Decide is one example of the Goal-based Scenario (GBS) [5] framework for learn-by-doing environments. The purpose of GBS is to engage students in active learning [1]. A previous version of Indie was developed in Lisp on the Macintosh by the Institute of Learning Sciences (ILS) at Northwestern University. Over a dozen of Investigate and Decide learning environments were built with the old Indie (see [3]). Our new version is Java-based for portability and web-delivery, and uses XML files to represent the domain content. In addition, the new version adds much more complex tests, random test result generation, and time and cost calculations. Also, the new Indie supports human critiquing. Since authoring critiquing rules was the hardest part of the previous Indie, offloading this on a human expert makes scenario construction significantly easier.

Recently and independently of ILS, Chee *et al.* built a GBS system to teach diagnosis of fetal abnormalities [2]. The major difference between their system and Corrosion Investigator is that Chee's system allows students to make a diagnosis without providing supporting evidence, whereas Corrosion Investigator requires students make a well-supported claim with evidence. Having students explicitly present their evidences in the report provides an opportunity for the system and the instructor to examine students' underlying understanding and make critiques on it.

See [3] for more related work.

Discussion and Future Work

The input for Indie are XML files that describe the content information. XML files are in plain text files and editing them requires no programming skills, but they are very inconvenient for authors who are non-programmers to edit directly. We are developing a content authoring tool to provide a user-friendly interface and validation of input values, using a class-based dialog box approach that proved quite successful in the previous Indie.

In addition, we are continuing interviewing the instructor in the biofilm course to identify more features needed in Corrosion Investigator and therefore enhance Indie's functionality. Both Indie and Corrosion Investigator are still works in progress.

Conclusion

In summary, we have described Indie, a software engine for building Investigate and Decide environments. Indie reads in XML files that contain content knowledge and outputs an Investigate and Decide learning environment. In an Investigate and Decide learning environment, students run tests and build arguments. Reports submitted by the students will be sent to the instructor for critiquing.

Indie is a content-independent tool. While currently the learning environment that we are using Indie to build is in biofilm domain, it can create any Investigate and Decide learning environment according to the input files. We aim to have teachers in different domains use Indie to build various Investigate and Decide learning environments.

Acknowledgements

This work was supported primarily by the Engineering Research Centers Program of the National Science Foundation under Award Number EEC-9876363. Matthew Parsek, assistant professor in Civil Engineering, developed the original course, and then expanded and refined the content for Corrosion Investigator. Ann McKenna, a post-doctoral fellow in the School of Education and Social Policy, guided the development and evaluation of the original course.

References

[1] J. D. Bransford, A. L. Brown, & R.R. Cocking (Eds) (1999*). How people learn: Brain, Mind, Experience, and School.* Washington, DC. National Academy Press.

[2] Chee, Y. S., Sosa, R., Tham, E., Sng, S. S., Choolani, M., Biswas, A., & Lun, K. C. (1999). *Multimedia goal-based scenario for learning to diagnose fetal abnormalities.* In Proceedings of ED-MEDIA 1999-World Conference on Educational Multimedia, Hypermedia & Telecommunications, Seattle, WA, USA, pp. 846-851. Charlottesville, VA: Association for the Advancement of Computing in Education.

[3] Dobson, W.D. 1998. *Authoring Tools for Investigate and Decide Learning Environments.* Ph.D. thesis.

[4] Indie. http://www.cs.northwestern.edu/~riesbeck/indie/

[5] Shank, R., A. Fano, B. Bell, and M. Jona. 1993. *The Design of Goal-Based Scenarios*. Journal for the Learning Sciences 3:4. 305-345.