# Open Goal-Based Scenarios: An Architecture for Hybrid Learning Environments

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

**Abstract**: We describe a software toolkit called Indie, for building hybrid Investigate and Decide (Indie) learning environments. In Indie learning environments, students can run simulated experiments, construct reports arguing for different claims, and submit these reports for critiquing. The Indie learning environment is one example of the Goal-based Scenario (GBS) learn-by-doing environments. Previous examples of GBS systems have been self-contained software-based learning environments. The content of the learning environments is always pre-defined, which leaves little space for open-ended activities. We overcome this problem by creating an architecture for hybrid GBS's that provide the option for having human coaching in the learning environments. This brings the possibility of having open-ended activities in GBS's as well as the opportunity for incrementally augmenting the content at instruction time. Indie has been used to construct Corrosion Investigator, a learning environment for engineering undergraduates in the domain of biofilms.

## Introduction

The Goal-based Scenario (GBS) (Schank, 1994; Schank, Fano, Bell, Jona, 1993) framework for computer-based learn-by-doing (Dewey, 1916) environments is designed to engage learners in active purposeful learning (Bransford, Brown, Cocking, 1999). A GBS begins by giving learners some large realistic problem that needs to be solved. With support from the learning environment, students learn target skills and underlying concepts in pursuit of goals within a context rich environment. The content of a GBS system is authored in advance based on real-life cases, common pitfalls, war stories, and so on. GBS's provide not only a simulated world for interaction, but also modules for expert coaching and critiquing, so that the learners are not limited by the availability of domain experts.

In order to support such computer-based coaching and critiquing modules, GBS's have to be closed world systems. The vocabulary of operations and situations has to be specified in advance so that rules can be written to provide accurate and timely feedback. Once deployed, learners can only do what the GBS has been prepared to support, so designers have to anticipate every action a student might want to do, and the most appropriate feedback for such actions in all situations. GBS's require significant upfront design, implementation, and piloting by content developers working with domain experts and test subjects, in order to provide learners with a rich challenging environment. There is no easy way to add new operations, coaching, or critiquing later in order to incorporate new developments in the domain or locally relevant content. Furthermore, it is difficult for GBS systems to accommodate the interests of open-ended teaching and learning paradigms (Collins, Brown, Newman, 1989; Gardner, 1991; Papert, 1993).

To overcome the above difficulties, we have been creating an architecture for hybrid GBS's and applying it to a subclass called Indie (Dobson, 1998). In Indie learning environments, students learn how to solve problems by running experiments and using the results to support or refute possible diagnoses. A hybrid Indie learning environment does not only have traditional support in a GBS from the computer, but also allows human experts in the feedback loop to handle totally open-ended inputs from the students that previously are not allowed and understood by the system. Open-ended activities are therefore made possible with the help from the human expert. Meanwhile, new material can be introduced in the interactions by students and faculty during the running of the GBS. There no longer exists a point where the system content is frozen. Introducing a human expert into the learning environment makes both the learning environment and the development of the learning environment open.

**Indie Architecture**

Indie is a content-independent Java-based software toolkit that includes a content authoring tool and a runtime learning environment (Fig. 1). The content authoring tool provides a user-friendly knowledge input interface for experts to enter content knowledge into a knowledge base, e.g. scenario challenge, experiment options, result generation methods, and domain background information. The Indie Learning environment reads in content files and become instantiated to a web-based learning environment. The Indie learning environment has 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. Indie learning environments automatically generate lab test results based on requests from students and provide scaffolding for students to construct arguments. Working as middleware between the students and instructor, it tracks student working progress for the instructor and lets the instructor role-play characters in the scenario.

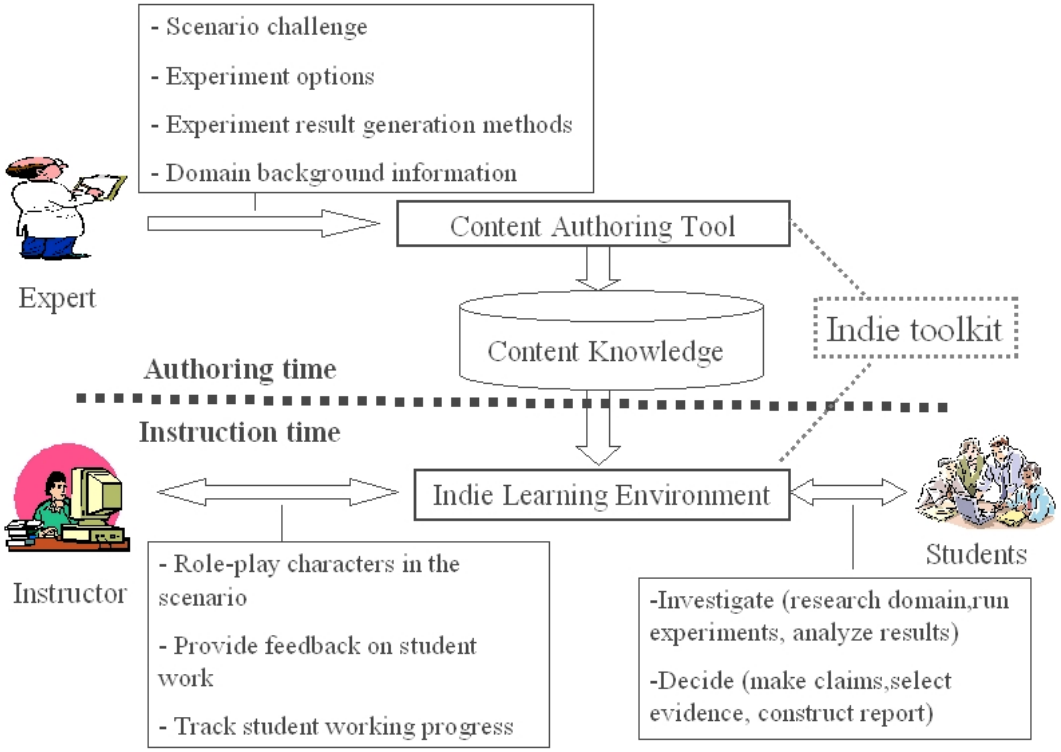


**Figure 1**: Architecture of the Indie toolkit

Though authoring time and instruction time in Figure 1 respectively refers to the time when the expert inputs knowledge into the system and the time when the instructor works with the students through the learning environment, they are not necessary separate periods. In other words, the instructor can work as the expert to augment the system while engaging in working with the students. Having the instructor working in the learning environment keeps the system from totally depending on pre-defined content, which is crucial when the content knowledge in the system is still under development.

**An Example: Corrosion Investigator**

We used Indie to build the Corrosion Investigator application, a learning environment for engineering undergraduates in the domain of biofilms. In the learning environment, students take the role of consultants to help a company determine the cause of pipe corrosion. Students need to make conscious decisions about which tests to run and which test results support the claims they make in order to solve the problem in a timely and economical manner. This requires students fully understand the purposes of the tests and the implication of the

test results.

To run tests, students go to the Experiment page (Fig. 2). Here, students can 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. Currently, there are five tests in Corrosion Investigator. A few more will be added. If no test matches the name that students enter, students will be asked to try another name, or email the Scientific Consultant, a character played by the instructor in the scenario, for help. Using an open-ended interface for test names forces students to research and brainstorm about what tests might be relevant. It also allows tests to be used that the scenario authors hadn't predicted. Hence, the problem solving activity is more open-ended and closer to real life situations. Such activities help students develop adaptive expertise (Bransford et al., 1999). New tests and results can then be incorporated into the system for reuse by future students. This approach does require an expert in the loop, though, to interpret reasonable but unexpected test names, and to generate plausible test results on the fly.

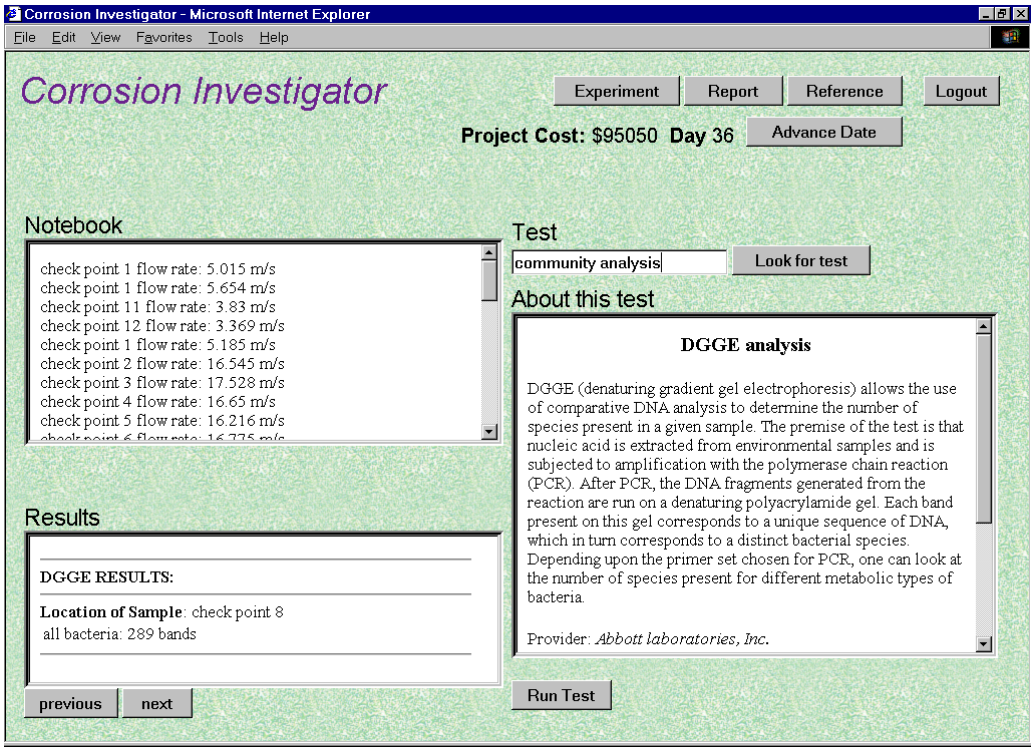


**Figure 2**: The Experiment screen in Corrosion Investigator.

When the test results become available, they appear in both the Notebook and Test Results area (Fig. 2). 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 for constructing a report. Most of the numbers in the test results are generated randomly based on constraints specified in the domain files.

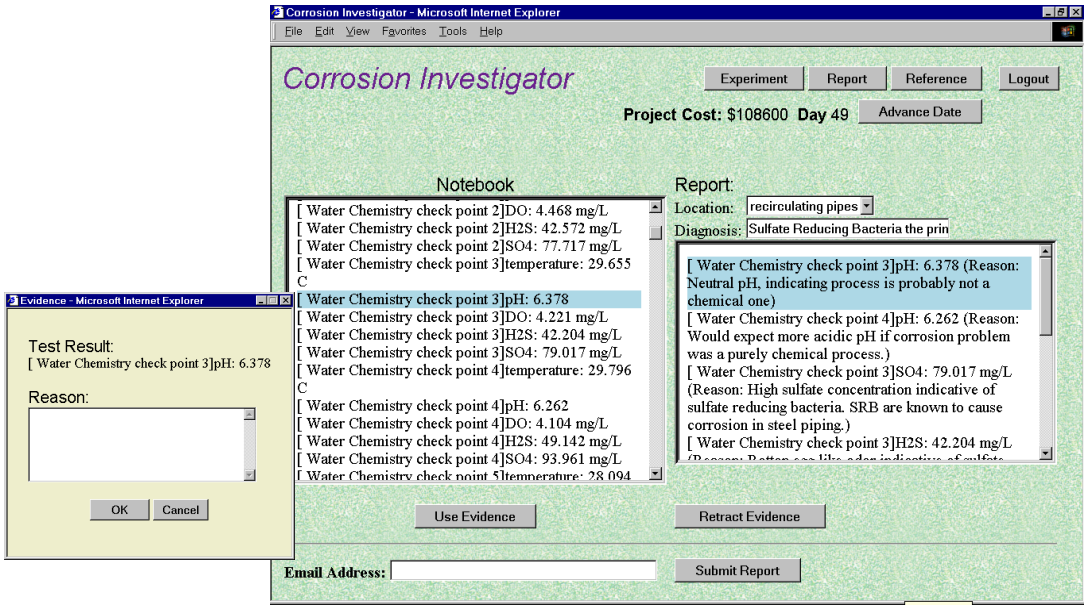


**Figure 3**: The Report screen in Corrosion Investigator.

When students feel they have gathered enough information, they can go to the Report screen (Fig. 3), make claims, and apply evidence in the Notebook towards that claim. Students need to specify the reason for using a test result as evidence. They also 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. 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.

## Related Work

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 Dobson, 1998). Our new version is Java-based for portability and web-delivery, and uses XML files to represent the domain content. Most imporantly, the new Indie supports human coaching and critiquing as well as much more complex tests, and random test result generation.

See Dobson (1998) for more related work.

## Future Work

Six first-year graduate students in the Civil Engineering Department in Northwestern University used Corrosion Investigator in a class in May 2002. After using Corrosion Investigator for 3 weeks, students completed a survey regarding their experience. Their average score to the statement that "overall, this is an excellent system for doing the project" was 2.16 on a scale of 1 to 5 (1 stands for Strongly Agree, 5 stands for Strongly Disagree). The question "How difficult is it for you to find out the right test to run?" got the most negative score, 3.83 on a scale of 1 to 5 (1 stands for Very Easy, 5 stands for Extremely Hard). This shows that the most difficult aspect of the interface to use was the mechanism for finding tests to run. Improving the interface for finding tests to run thus becomes one of our future research issues. Other parts of the system were considered not difficult. As for the professor of the class, the software significantly reduced his work during the project phase, especially on test result generation.

The content knowledge for Corrosion Investigator was written in XML format directly without using the Indie authoring tool because the tool is still under development. We are using a class-based dialog box approach that proved quite successful in the previous Indie for building the tool.

**Acknowledgements**

This work was supported primarily by the Engineering Research Centers Program of the National Science Foundation under Award Number EEC-9876363. Matthew Parsek, an assistant professor in Civil Engineering, developed the original biofilm course that Corrosion Investigator was based on. He later expanded and refined the content in his course 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**

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

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*, Lawrence Erlbaum Associates, Hillsdale, NJ

Dewey, J. (1916) *Democracy and Education*, The Free Press, New York.

Dobson, W.D. (1998). Authoring Tools for Investigate and Decide Learning Environments. *Ph.D. thesis*.

Gardner, H. (1991) *The Unschooled Mind, How Children Think & How Schools Should Teach*, BasicBooks, New York.

Papert, S. (1993) *Children's Machines: Rethinking Education in the Age of the Computer*, BasicBooks, New York

Schank, R. (1994). Goal-based scenarios: *A radical look at education*. Journal of the Learning Sciences 3, 4, 429-453.

Schank, R., Fano, A., Bell, B., & Jona, M. (1993). The Design of Goal-Based Scenarios. *Journal for the Learning Sciences* 3:4. 305-345.