

# The Object-Oriented Design Learning Environment

**OODLE**  
**Stottler Henke**  
Smarter Software Solutions

The Object-Oriented Design Learning Environment (OODLE) is a prototype Intelligent Tutoring System funded by the National Science Foundation to teach high-level software engineering—in particular, object-oriented design. The object-oriented approach continues to sweep the software industry, becoming the dominant approach to large-scale, adaptable, and reusable systems development. OODLE is targeted at current college students, as well as the many software professionals whose training and experience were dominated by pre-object technology.

Intelligent Tutoring Systems offer individualized instruction without the need for a human instructor. OODLE's pedagogical approach is based on the work of Craig Larman, author of the best-selling text *Applying UML and Patterns*. It offers students extensive supervised experience in making key object design decisions, while emphasizing the principles and rationale that underlie well-informed decisions.

OODLE challenges the student with a progressive set of realistically complex design problems, each broken down into a set of iterative design cycles. For instance, the first problem asks the student to design

software that implements the rules of the Monopoly? game. Figure 1 shows part of the problem set up.

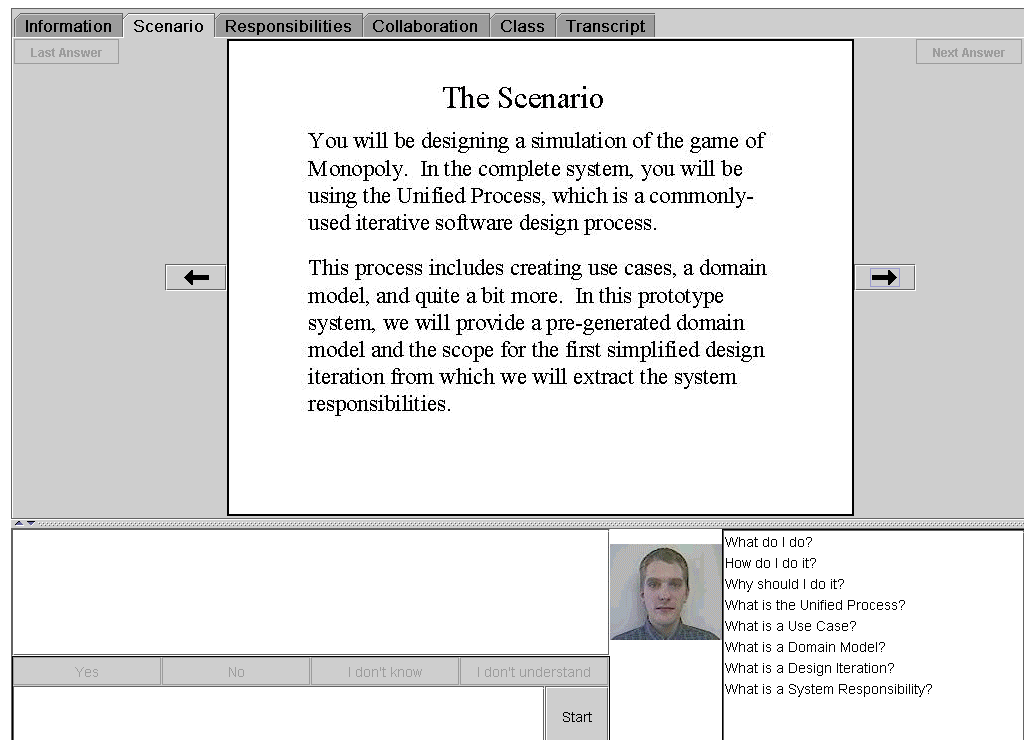
The first Monopoly iteration focuses on simple turn management: the student's system must manage a set of players taking turns, rolling dice, and moving around the board. Just as in real-world iterative design, features such as properties, special squares, and money, are added in later iterations.

OODLE emphasizes *justified design decisions*. Design is not about inspiration or some loose notion of esthetics. Larman's tested approach to object-oriented design relies on rational justifications for why some design options are better than others in certain situations. OODLE provides extensive practice in making and justifying design decisions on a rational basis.

Stottler Henke is currently seeking input from organizations that have a stake in improved methods and tools for teaching object-oriented design. We are also seeking follow-on funding from NSF to develop a robust, feature-rich version of the system over the next one to two years.

FIGURE 1. THIS INFORMATION SCREEN DESCRIBES THE FIRST DESIGN PROBLEM IN THE OODLE SYSTEM. THE TUTOR IS PICTURED AT THE BOTTOM. ITS REMARKS APPEAR IN THE PANE TO ITS LEFT, WHILE STUDENT RESPONSES ARE INPUT IN THE PANE BENEATH. QUESTIONS THE TUTOR IS PREPARED TO ANSWER APPEAR IN THE LIST TO THE RIGHT OF THE TUTOR.

THE TABS ACROSS THE TOP OF THE SCREEN PROVIDE ACCESS TO SEVERAL DIFFERENT VIEWS, INCLUDING COLLABORATION AND STATIC CLASS UML DIAGRAMS, A RESPONSIBILITY ASSIGNMENT TABLE, ANSWERS TO STUDENT QUESTIONS, AND A TRANSCRIPT OF THE ENTIRE STUDENT/TUTOR INTERACTION.



## The Responsibility Assignment Cycle

Each iterative design cycle for each design problem starts with a presentation of the goals and constraints of that iteration, and then proceeds through a series of decisions the student must make. The current system emphasizes *responsibility assignment decisions*, as these are the most fundamental aspect of object-oriented design. Figure 2 shows the screen as the student considers possible justifications for an early design decision.

## Mentored Experience

Object-oriented design skills are typically picked up through formal classes, self-study, or time-consuming on-the-job experience. But everyone knows a class, a seminar, or a book does not create a practiced expert. And in the real world, getting practice on-the-job is expensive in terms of lost productivity—not just the

apprentice designer, but also managers and mentors lose time and risk ending up with a faulty product.

This is why the intelligent tutoring in OODLE is so important: extensive guided practice across a range of problems is the kind of experience that makes for real expertise. By understanding the demands of the problem and the typical behaviors of students, OODLE can closely track student performance and offer specific targeted feedback, support, and instruction. And the OODLE intelligent tutor can provide that experience whenever and wherever the student wants, without tying up other valuable personnel.

Figure 2 illustrates the system critiquing the student's chosen rationale for the responsibility assignment in question. OODLE prompts the student to offer an alternative rationale, or the student may proceed to try an alternate assignment entirely.

FIGURE 2. THE RESPONSIBILITIES TAB PROVIDES A CONTEXT IN WHICH STUDENTS CAN WORK THROUGH THE MOST FUNDAMENTAL DESIGN DECISIONS THAT ARISE IN OBJECT-ORIENTED DESIGN: WHICH CLASSES ARE GOING TO BE RESPONSIBLE FOR HANDLING WHICH RESPONSIBILITIES? IT IS NOT ENOUGH TO CHOOSE A CLASS TO HANDLE A RESPONSIBILITY (EVEN A REASONABLE CLASS); THE STUDENT ALSO HAS TO OFFER A SOLID JUSTIFICATION FOR WHY THAT CLASS SHOULD HANDLE THAT RESPONSIBILITY. THE TUTOR IS PREPARED TO DISCUSS STUDENT CHOICES AS A WAY TO BUILD DEEP UNDERSTANDING OF THE DECISIONS AND PRINCIPLES AT STAKE.

Project Monopoly Game		Responsibility
Responsibilities	Candidates	Justifications
Initiate Game Play	<input checked="" type="radio"/> MonopolyGame	Controller
Each Player Take Turn	<input type="radio"/> Player	

While a Player would probably know how to take its own turn, it probably does not know about the other players, which would be necessary to make sure that every player took a turn. Adding a reference to other players in the player object would greatly increase coupling. I'm not sure that that justification will work here. Are there any other justifications that would?

Yes No I don't know I don't understand

A justification for this responsibility is **Low Representational Gap** Say It

- What do I do?
- How do I do it?
- Why should I do it?
- What is Object Coupling?
- What are the disadvantages of high coupling?
- What are the benefits of low coupling?
- What are some common forms of coupling?

## Questions, First and Foremost

People learn best when they are actively seeking information—when they have questions. Problems in OODLE not only provide a context for practice, they also provide a motivation for learning. And OODLE proactively offers to answer questions that naturally arise in the course of ongoing design activity—questions that directly address the student's ability to solve the problem.

Whenever the system offers information to the student—be it during a discussion or a decision-making sequence—the system also gives the student a chance to ask relevant questions. The student can choose from among predefined questions likely to be raised by the latest material presented. Since the answer to a question can itself raise additional questions, a network of these predefined questions and answers runs throughout the system. Figure 1 and Figure 2 both show a sample of such context-

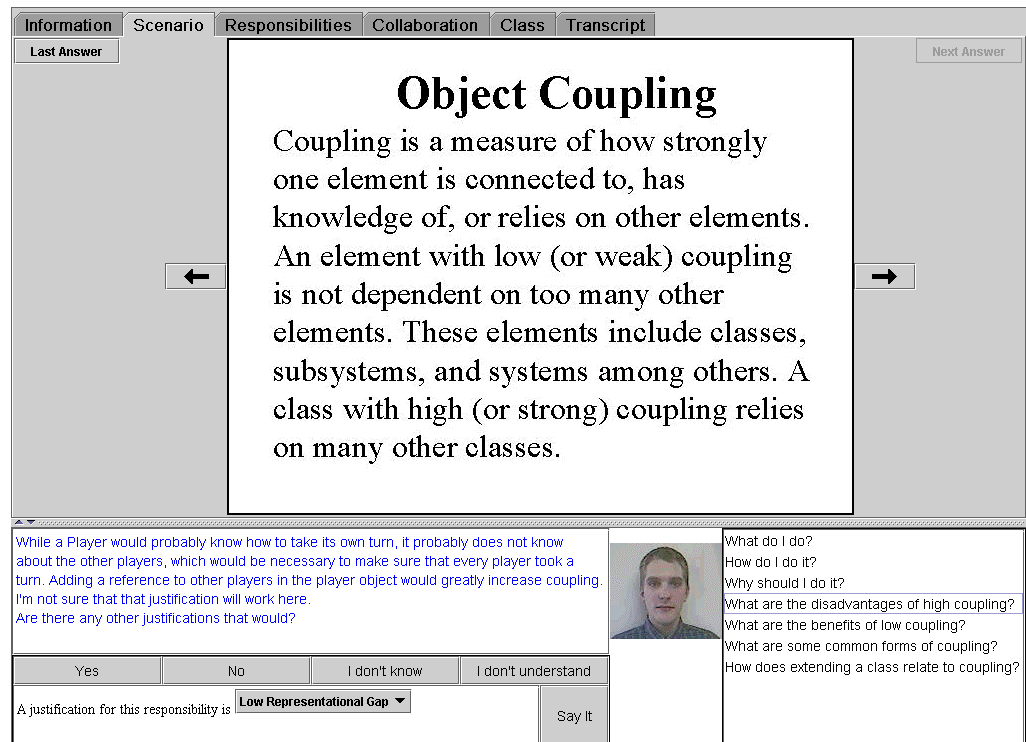
relevant questions in the lower right corner of the screen. Figure 3 shows a typical answer screen, with additional follow-up questions listed in the same position. The question and answer network allows students to recognize gaps in their understanding and bridge them.

### Understanding vs. Notations and Tools

The skill of object-oriented design is not the same as the skill of working with any particular object-oriented programming language or environment. Despite the proliferation of tools and notations in

general, there is a growing consensus that the Unified Modeling Language (UML) is the preferred notation for expressing object-oriented designs. Accordingly, OODLE adopts and exploits UML as a way to document the student's design in progress. However, the emphasis in OODLE is not on teaching UML—it is on teaching design skills. Figure 4 and Figure 5 illustrate how the system currently provides UML design diagrams to help the student visualize object-oriented designs. In future versions, the system will support a wider range of UML diagram types, potentially including sequence, domain, and use-case diagrams.

FIGURE 3. THE ASK TAB PROVIDES A WINDOW INTO A NETWORK OF INTERLINKED QUESTIONS AND ANSWERS THAT CAPTURE STUDENTS' MOST FREQUENTLY ASKED QUERIES.



### OODLE's Place in the World of Training

Unlike instructor-led classroom training (ILT) and traditional computer-based training (CBT) that introduce object-oriented facts and concepts and test their recall and application in simple exercises, OODLE lets students apply their knowledge and skills to solve realistically-complex design problems and receive automated, cost-effective individualized tutoring.

OODLE can be used in combination with ILT and CBT to reinforce concepts and achieve higher proficiency levels that are unattainable through passive learning. Or, OODLE can be used as a self-contained instructional system that presents object-oriented concepts just-in-time, in the context of solving hypothetical design problems.

### The Future of the Family

The current version of OODLE is a research prototype developed with seed funding from the National Science Foundation. It is aimed at an introductory level, for students who have used object-oriented programming languages and environments, but who have not been introduced to the principles and practices of good object-oriented design.

The operational version will include more problems for the student to work on and more appropriate tutoring. With the additional problem material, it will also support more decision types, and more forms of visualization to support those decisions. Socratic dialogue technology currently under development at Stottler Henke will enable yet higher levels of

interaction, including sophisticated discussion of design alternatives.

Future modules will cover intermediate and advanced object-oriented design, including the application of design patterns and architecture-level design issues. But at each level of instruction, OODLE will present a progressive series of complex problems to provide extensive, varied, realistic, mentored practice designing object-oriented software.

The same proprietary technology can be used to build specialized versions of OODLE at any level to focus on more specific application niches (e.g. banking, insurance, web applications, etc). The effectiveness of problem-based learning is increased when the problems are closer to those faced by students in their work lives.

## About Stottler Henke

Stottler Henke Associates, Inc. ([www.shai.com](http://www.shai.com)) develops and deploys innovative, practical software systems that apply artificial intelligence (AI) technologies to solve problems that defy solution using traditional approaches. A decade of work on Intelligent Tutoring Systems (ITS) has placed the firm at the forefront of a technology that is poised to remake the way people learn everything from common school subjects to specialized technical and professional skills. Stottler Henke ITSs are at work today, helping customers such as the US military meet their toughest training challenges.

FIGURE 4. THE COLLABORATION TAB SHOWS A SET OF STANDARD UML COLLABORATION DIAGRAMS BUILT BY THE SYSTEM IN RESPONSE TO THE STUDENT'S DECISIONS.

The screenshot shows the 'Collaboration' tab in the OODLE interface. At the top, there are navigation tabs: Information, Scenario, Responsibilities, Collaboration (selected), Class, and Transcript. The main area displays a UML Collaboration Diagram with a message arrow labeled 'playGame()' pointing to a box labeled 'MonopolyGame'. Below the diagram is a student justification interface. It includes a text area with the following text: 'While a Player would probably know how to take its own turn, it probably does not know about the other players, which would be necessary to make sure that every player took a turn. Adding a reference to other players in the player object would greatly increase coupling. I'm not sure that that justification will work here. Are there any other justifications that would?'. Below this text are four buttons: 'Yes', 'No', 'I don't know', and 'I don't understand'. A dropdown menu is set to 'Low Representational Gap'. To the right of the justification area is a video feed of a student and a list of questions: 'What do I do?', 'How do I do it?', 'Why should I do it?', 'What are the disadvantages of high coupling?', 'What are the benefits of low coupling?', 'What are some common forms of coupling?', and 'How does extending a class relate to coupling?'. A 'Say It' button is located at the bottom right of the interface.

FIGURE 5. THE CLASS TAB SHOWS A SET OF STANDARD UML STATIC CLASS DIAGRAMS BUILT BY THE SYSTEM IN RESPONSE TO THE STUDENT'S DECISIONS. AS NEW CLASSES ARE INTRODUCED INTO THE DESIGN THEY APPEAR ON THIS PAGE. AS NEW METHODS AND FIELDS ARE ASSIGNED TO A CLASS, THEY TOO APPEAR IN THE DIAGRAM.

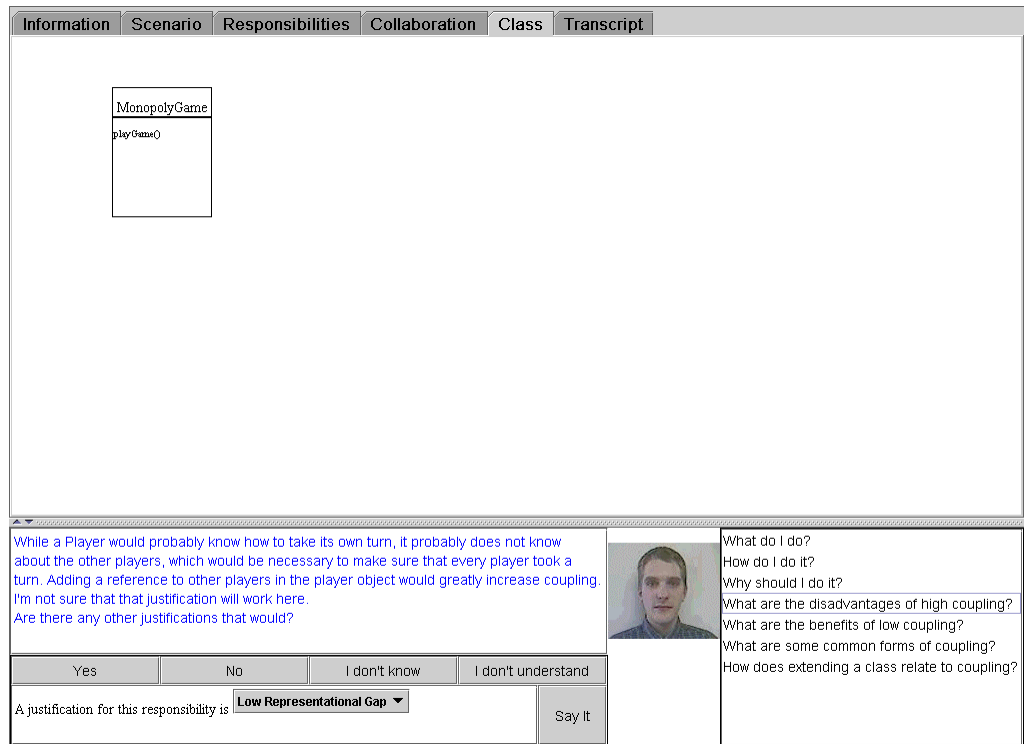


FIGURE 6. THE TRANSCRIPT TAB PROVIDES ACCESS TO A COMPLETE HISTORY OF THE STUDENT'S INTERACTION WITH THE TUTOR ON A GIVEN PROBLEM.

