Experiment Centered Design in a Massively Multiplayer Online Educational Game

by

Shawn Conrad

B.S., Massachusetts Institute of Technology (2011)

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

February 2013

© 2013 Massachusetts Institute of Technology. All rights reserved.

Signature of Author ...........................................................

Department of Electrical Engineering and Computer Science

February 1, 2013

Certified by ........................................................................

Eric Klopfer
Professor of Urban Studies and Planning
Thesis Supervisor

Accepted by ........................................................................

Dennis M. Freeman
Professor of Electrical Engineering and Computer Science
Chairman, Masters of Engineering Thesis Committee
ABSTRACT

With the United States of America suffering from a lack of scientifically literate grade and secondary school students, educational games offer an opportunity to engage and inspire students to take interest in science, technology, engineering, and mathematical (STEM) subjects. Learning assessment techniques coupled with machine learning algorithms can be utilized to record student’s in-game actions and formulate a model of the student’s knowledge. This paper describes “Experiment Centered Assessment Design” (XCD), a framework for structuring a learning assessment feedback loop. XCD builds on the “Evidence Centered Assessment Design” (ECD) approach, which uses tasks to elicit evidence about a student and his learning. XCD defines every task as an experiment in the scientific method, where an experiment maps a test of factors to observable outcomes. This XCD framework was applied to prototype quests in a massively multiplayer online (MMO) educational game. Future work would apply machine learning techniques to the information captured from XCD to provide feedback to students, teachers, and researchers.

Thesis Supervisor: Eric Klopfer
Title: Professor of Urban Studies and Planning
ABSTRACT - With the United States of America suffering from a lack of scientifically literate grade and secondary school students, educational games offer an opportunity to engage and inspire students to take interest in science, technology, engineering, and mathematical (STEM) subjects. Learning assessment techniques coupled with machine learning algorithms can be utilized to record student’s in-game actions and formulate a model of the student’s knowledge. This paper describes “Experiment Centered Assessment Design” (XCD), a framework for structuring a learning assessment feedback loop. XCD builds on the “Evidence Centered Assessment Design” (ECD) approach, which uses tasks to elicit evidence about a student and his learning. XCD defines every task as an experiment in the scientific method, where an experiment maps a test of factors to observable outcomes. This XCD framework was applied to prototype quests in a massively multiplayer online (MMO) educational game. Future work would apply machine learning techniques to the information captured from XCD to provide feedback to students, teachers, and researchers.

I. INTRODUCTION
A. Advantages of a Massively Multiplayer Online Role Playing Game for Education
   1. Open World Role Playing Games
   2. Massively Multiplayer Online Games
   3. Feedback in Massively Multiplayer Online Role Playing Games
B. “The Radix Endeavor”
C. Learning Assessment
   1. Evidence Centered Design
      i. Student Model
      ii. Evidence Model
      iii. Task Model
      iv. Assembly Model
      v. Presentation Model
      vi. Delivery System Model
   2. Assessment Cycle
   3. Back-end Assessment
4. Stealth Assessment

II. RELATED WORK
   A. World of Warcraft in School
   B. River City Project
   C. Newton’s Playground
   D. Quest Atlantis

III. EXPERIMENT CENTERED DESIGN (XCD)
   A. Structure of a Quest
      1. Prompt
      2. Experimentation
         i. Hypothesis
         ii. Test
         iii. Observations
      iv. Capturing the Scientific Method
      3. Solution
      4. Learning Concepts
      5. Feedback
         i. Player Feedback
         ii. Educator Feedback
      6. Information Flow
   B. Object Models
   C. Quest Template Object Model
      1. Quests
      2. Educational Content
      3. Experiments
      4. Users
      5. Experiment Series
   D. Centering Around the Experiment
      1. Extending Traditional Assessment in the ECD Framework
         i. Traditional Closed Response
         ii. Traditional Open Response
iii. Converting from Object Model to Database Storage

2. Experiment Centered Design and Related Work

3. Unique Features of Experiment Centered Design
   i. Parallel Development of Experiments
   ii. Experiment Series Capture Students' Work
   iii. Randomness in Experiments

IV. RESULTS

A. Quest Dialog

B. Volume and Surface Area
   1. Quest Object Models
      i. Game Designer Object Model
      ii. Researcher Object Model
      iii. Merging Object Models into Database Storage
   2. Digital Prototype
      i. Experiment Emulation
      ii. Data Flow and Back-end Storage
   3. Generalization

C. Mendelian Inheritance
   1. Quest Object Models
      i. Experiment Object Models
      ii. Learning Concept Object Models
      iii. Player Object Models
         a. Player Alice
         b. Player Bob
         c. Player Charlie
   2. Digital Prototype
      i. Experiment Emulation
      ii. Data Flow and Back-end Storage
      iii. Adjusting Student Model from Experimentation
         a. Probability that Red Color Trait is Dominant Given Experiment One
            Occurs
1. Probability that Experiment One Occurs Given Red Color Trait is Dominant
   I. Probability of Selecting Parent Flower Genotypes
      A. Flowers Selected from a “Stable State” Garden
      B. Flowers Selected with Equal Probability
   2. Probability that Experiment One Occurs Given White Color Trait is Dominant
   3. Probability that Red Color Trait is Dominant Given Experiment One Occurs as First Experiment
   b. Visualizing Student Model Updates Over Time

3. Generalization

V. FUTURE WORK
   A. Experiment Centered Design in Future Endeavors
   B. Classification of Experiments in Experiment Centered Design
   C. Social Experiments in Experiment Centered Design
   D. Feedback Portals

VI. CONCLUSION

I. INTRODUCTION

The United States of America is currently suffering from the lack of a scientifically literate student population. An understanding of science, technology, engineering, and mathematical (STEM) subjects is vital for entering tertiary education and the skilled labor force. Without a solid grounding in STEM subjects, U.S. citizens face a decline in scientific and industrial influence that is already threatened by other countries. One method for improving STEM learning is to foster an interest in conducting and understanding science. By promoting the epistemology of science, one hopes to engage and develop student abilities in STEM subjects. (Klopfer, 2011)

The Education Arcade (TEA) is a research group at the Massachusetts Institute of Technology that promotes learning in primary and secondary school students through games that teach and engage. TEA has adapted popular commercial games as well as developed games in-
house to teach subjects like mathematics, biology, chemistry, and literature. TEA’s mission statement is “to demonstrate the social, cultural, and educational potentials of videogames by initiating new game development projects, coordinating interdisciplinary research efforts, and informing public conversations about the broader and sometimes unexpected uses of this emerging art form in education.” (Hass, 2008)

A. Advantages of a Massively Multiplayer Online Role Playing Game for Education

The Education Arcade believes that games are an effective medium to engage students in learning for many reasons. Primarily, games offer an alternate reality wherein the rules are set by the game designers. The rules and features of this game environment define a “semiotic domain.” Users are taught the rules and features by playing the game, and the game can follow particular principles that encourage learning. For example, by providing players with plenty of lives and frequent save points, the player is less likely to become discouraged after losing one life in the game. This ability for players to take risks and fail with minimal real-world consequences is called the “psychosocial moratorium principle”. By tying game principles that encourage exploration to challenges that promote learning an educational subject, an educational game offers a fun, safe environment to learn. (Gee, 2007)

I. Open World Role Playing Games

The rules that govern the interaction model between users and a game define the game’s genre. A role-playing game (RPG) is characterized by the player assuming the role of an in-game character, who they often customize and upgrade with various traits, equipment, and skills. An open-world, or “sandbox,” RPG allows players to move and act freely within the game environment, instead of following predefined paths and action sequences. Examples of open-world RPGs include “Final Fantasy” by Square Enix, “Diablo” by Blizzard Entertainment, and “Pokémon” by Nintendo. (International Hobo Ltd., 2009)

The customization and freedom offered by sandbox RPGs creates a semiotic domain that fosters growth and exploration. A game that allows users to identify with their in-game character (or “avatar”) follows the “identity principle.” The user associates the growth of his character
with his own personal growth. A game that allows multiple solutions to challenges follows the “multiple routes principle.” Players are encouraged to explore different options in an attempt to find a solution to a problem. Sandbox RPGs are well suited for multiple learning principles. (Gee, 2007)

2. Massively Multiplayer Online Games

A massively multiplayer online (MMO) RPG is a genre of games that allows hundreds to thousands of users to role play simultaneously. The most popular and successful MMORPGs is Blizzard Entertainment’s World of Warcraft (WoW), wherein thousands of players can explore, fight, craft, and communicate. While it is possible to play without interacting with other players, the most rewarding accomplishments and equipment often require cooperation with other players. MMORPGs thrive on this social interaction. (Blizzard Entertainment Inc., 2012)

Goals that require teamwork need people with good communication and cooperation skills. These goals, or “quests,” may require different types of cooperation. For example, in order to slay a mighty monster, a team or “party,” of characters may need to balance offense and defense to survive a difficult battle. In another quest, a user may need help from other players to find the solution to a difficult puzzle. The “affinity group principle” applies to players who share the same goals or experiences while playing the game. The “distributed principle” follows from players distributing information about the game world. Both of these principles are applicable to players of an MMORPG. (Gee, 2007)

3. Feedback in Massively Multiplayer Online Role Playing Games

MMORPGs have a multitude of information to convey to players about their characters’ status, abilities, location, equipment, enemies, and achievements. All of this information informs the user about his character and progress, and can be presented in a variety of ways through graphical user interfaces (GUIs). Information about a player’s immediate state such as health, location, and available actions are often shown in a heads-up display (HUD) overlaid on the display of the character in his environment. Reference information including quest logs and game settings are often displayed through in-game menus. Information about a player’s
achievements may exist on a separate, public interface to broadcast them to other players. And some information may appear in different forms across different interfaces, such as a player’s inventory being accessible from the HUD or a menu. All of these interfaces offer different levels of granularity about the player and environment for different purposes.

Most importantly, these interfaces allow a player to witness and analyze his position in the game. By efficiently tracking and displaying a user’s state, the game empowers users with the ability to make informed choices on the most effective path to success. Even with efficient user feedback, however, a player may still become stuck. The ability to detect when a player is stuck allows a game to follow the “explicit information on-demand and just-in-time principle.” Players should only be given information as needed or when it is most applicable. When a player has become stuck, the game should gradually assist the player in reaching his goal. By trusting and respecting a player’s ability to progress with minimal help, the game preserves an environment that encourages exploration and learning. (Gee, 2007)

B. “The Radix Endeavor”

The Education Arcade is actively developing a massively multiplayer online role playing game to take advantage of these and other learning principles. “The Radix Endeavor” is a sandbox MMORPG set on a mysterious cluster of islands. The people of these islands live in a time reminiscent of the Middle Ages, when science and technology were limited. Furthermore, the government suppresses the population’s ability to practice science in order to maintain control over them. Players are recruited to a secret society that defies the government and is finding scientific discoveries to undermine and overthrow the reigning regime. (Klopfer, 2011)

The learning goal of “The Radix Endeavor,” or “Radix” for short, is to engage high school students in learning mathematics and biology. Students assume different character roles that determine the curriculum of quests their character will need to complete. These quests have a range of difficulty and can occur in various locations across the islands. In addition, these quests may be solitary or require multiple players to cooperate.

In the majority of quests, the key to solving the task requires an educational insight. For example, a player may need to understand the concept of similar triangles in a mathematics quest or Mendelian inheritance in a biology quest. Because of the sandbox MMORPG structure,
players have the freedom to experiment and explore to uncover these insights. Additionally, players can ask for or offer help to other players to share their knowledge of the concepts and quests.

As previously stated, good user interface design is vital for conveying a student’s state in the game. In order to use “The Radix Endeavor” in a classroom setting, teachers will also require a user interface that shows a student’s game state and how this state reflects understanding of the core material. In addition, the teacher may require aggregation of his students’ progress so he can understand where his students are struggling and adjust his lesson plan accordingly.

C. Learning Assessment

The ability to record, summarize, display, and improve a player’s progress requires learning assessment. Assessment measures a user’s understanding of his situation, forms a model of that student’s knowledge, and acts on this model to further the user’s goals. The techniques used to formulate this model and offer recommendations are an active area of research. (Shute & Ventura, 2013)

The traditional form of learning assessment occurs in the classroom. A teacher teaches his students and measures their knowledge by administering tests. This cycle is limited by the validity of the tests and the teacher’s competence at interpreting the results. The validity of a test is its ability to accurately measure a student’s knowledge, and tests with poor validity misrepresent students’ abilities. Poor interpretations lead a teacher to under- or overcorrect lesson plans for his students. Furthermore, infrequent testing allows students’ weaknesses to go undetected and uncorrected for extended periods of time. Digital assessment can benefit teachers and students by offering tighter feedback loops that correlate a student’s performances with their academic strengths and weaknesses. (Shute & Ventura, 2013)

1. Evidence Centered Design

Evidence Centered Assessment Design (ECD) is an approach to constructing educational assessments that focus on measurable evidence of a student’s learning. ECD collects and analyzes evidence from tasks performed by the student. Collectively, these student, evidence,
and task models form the Conceptual Assessment Framework (CAF). Refer to Figure 1 below. The following paragraphs briefly describe each of the CAF models and how they could be applied to traditional classroom learning assessment. (Mislevy, Almond, & Lukas, 2003)

![Delivery Model Diagram](image)

**Figure 1: Distinct models of the Conceptual Assessment Framework**

### i. Student Model

First, the student model is a representation of a student's knowledge of a particular domain. For this example, the student model represents a student's understanding of biology. While it is impossible to gain an exact depiction of a student's knowledge, one must approximate and represent this knowledge. Hence, the student model is traditionally represented by a numeric grade (0-100) and simplified into the letters A, B, C, D, and F. In short, the student model asks “what competency are we measuring?” (Mislevy, Almond, & Lukas, 2003)

### ii. Evidence Model

Second, the evidence model provides instructions on how the student model should be updated given the result of a student’s work on tasks. The evidence model has two parts. Evidence rules identify and summarize the meaningful work that shows evidence of learning.
The measurement model accumulates and interprets this meaningful work to update the student model. In this example, the evidence model is the comparison of a student’s test answers with the solutions. The rubric provides the evidence rules that label a student’s work as correct or incorrect. The weight and impact of the exam on the student’s grade (the student model) is the measurement model. In short, the evidence model asks “how do we measure competency?” (Mislevy, Almond, & Lukas, 2003)

iii. Task Model

Next, the task model is composed of the scenarios that can elicit the evidence needed to update the student model. In this example, the tasks are the questions on a particular exam, which could be multiple choice or open response. The student’s answers are the output of the task model and the input to the evidence model. In short, the task model asks “where do we measure one’s competency?” (Mislevy, Almond, & Lukas, 2003)

iv. Assembly Model

The assembly model structures the balance needed to gain an accurate student model from the family of tasks in the task model. In this example, a teacher must balance the question content and types before administering the test. The information obtained from an open response question may be more content rich than that of multiple-choice questions, and the teacher must determine what selection of questions is sufficient. In short, the assembly model asks “how much do we need to measure?” (Mislevy, Almond, & Lukas, 2003)

v. Presentation Model

The presentation model is defined by the medium that the tasks are delivered to the students. In this example, the exam may be administered with pencil and paper, through a computer interface, or even orally. The presentation model should not encumber the student and easily facilitate the assignment of tasks and collection of evidence. In short, the presentation model asks “how do the tasks look?” (Mislevy, Almond, & Lukas, 2003)
vi. Delivery System Model

Finally, the collection of the student, evidence, task, assembly, and presentation models define the delivery system model. This model is intended to capture any issues not previously described by another model, such as the timing or security of the system. In this example, it is likely that the test is time-constrained and students are separated to avoid cheating. In short, the delivery model asks “how does the system work?” (Mislevy, Almond, & Lukas, 2003)

2. Assessment Cycle

The ECD framework is intended to work within an assessment cycle defined by four key processes. This cycle defines the flow from selecting, displaying, performing, and scoring a task. Refer to Figure 2 below.

The cycle begins with a set of tasks being selected from a large library of tasks by an administrator. Tasks are presented to the participant to work on. Upon completion, the participant submits his responses for processing. Processing a response includes interpreting a student’s answer as well as any metadata captured from the student’s work. Feedback about particular tasks can be reported to the student, teacher, or other interested parties. A response is
also used to update the cumulative score of the user. Feedback that summarizes the student’s overall score can be pulled from this summary scoring process. The scoring process updates the model of the participant’s knowledge, and a new set of tasks is selected for the participant. The cycle may continue until a significant model is formed about the participant. (Mislevy, Almond, & Lukas, 2003)

In the traditional classroom example, the teacher selects all of the questions for a test ahead of time. The test presents these questions to students, which captures their work and answers. The teacher scores the tests, updates the students’ grades, and identifies which subjects to teach and review for the future.

While the example above applied ECD to a traditional classroom assessment, digital technology can enhance the process of selecting tasks and aggregating results. For instance, imagine individually presenting questions to a student, where each new question depends on the student’s answer to the previous question. This process could identify the student’s strong and weak subjects by dynamically avoiding topics that the student has mastered and focusing on questions that probe the student’s weaknesses.

3. **Back-end Assessment**

Aggregating results can leverage many advantages of digital technology as well. Data mining is the process of analyzing large amounts of logged data for trends and patterns. By applying data mining processes to evidence collected in ECD, students’ statistics and patterns can easily be brought to educators’ attentions. A variety of back-end assessments, including item response models and artificial neural nets, can offer various levels of insight and interpretation. (Quellmalz, et al., 2011)

4. **Stealth Assessment**

The ability to offer analysis and feedback to students as they perform tasks can help the students self-diagnose their strengths and weaknesses. However, one of the strengths of RPGs is their ability to actively engage students through fantastical roles and scenarios. A distinction between game tasks and assessment tasks would break this illusion and disengage students.
Stealth assessment integrates evidence of learning competencies into game tasks in the ECD framework. Hence, a student’s work on a task in-game reveals evidence about their knowledge of a particular subject. (Shute & Ventura, 2013)

II. RELATED WORK

Other massively multiplayer online role playing games have been used or built for educational purposes. Each of these projects shows positive correlation between students’ engagement and learning. The amount of data collected on students’ in-game activity varies depending on the goals of the project and affects the method of learning assessment.

A. World of Warcraft in School

First, the “World of Warcraft in School” initiative uses the off-the-shelf, commercial game World of Warcraft to teach subjects like math and literacy. Students learn these skills by coupling in-game quests with out-of-game activities. “WoW in School” provides a successful example of using an MMORPG to engage and teach students. (Gillispie, 2012)

Because WoW was not intended for an educational setting, there is little built in assessment for teachers to analyze a student’s progress. At best, the teacher can become privy to his students’ avatars and draw inferences from their quest logs and statistics. On the whole, teachers are responsible for manually defining, assigning, collecting, and analyzing assessments for each student’s work in the game. (Gillispie, 2012)

B. River City Project

The “River City Project” studied situated learning in a multi-user virtual environment (MUVE). Middle-school students assume the role of 19th century townsfolk who must investigate strains of illnesses plaguing the city. Avatars are capable of interacting with non-playable characters (NPCs), digital artifacts, experimental tools, and each other. In addition to playing the game, students complete out-of-game assignments from the teacher based on their experiences in the game. (Dieterle & Clarke, 2006)
The “River City Project” was intended to teach scientific inquiry. Hence, the project has a built-in system for logging students’ actions. Every significant artifact in the game is assigned an identification number. When a user clicks on a significant artifact, the interaction is associated with the student and saved to a log. Players’ movements throughout the game are also recorded. This “click-tracking” methodology leads to an overabundance of information. In addition, poor structural organization required this data to be parsed and interpreted by-hand. (Dieterle & Clarke, 2006)

C. Newton’s Playground

“Newton’s Playground” was a physics-based puzzle game that was built to demonstrate stealth assessment in games. The tasks required students to use simple machines like levers and ramps to move a ball towards a goal given various terrain and obstacles. The student models analyzed the players’ conscientiousness, creativity, originality, and physics intuition. The evidence for each of these tasks included the objects and the number of attempts used to solve the puzzle. (Shute & Ventura, 2013)

“Newton’s Playground” updates the student model after each completed puzzle. This model is presented to the teachers as feedback. Feedback for students, however, was left for future work. Providing students with feedback requires interpreting the student model and recommending adjustments per puzzle. Teachers could provide this service by interpreting students’ models manually. In addition, “Newton’s Playground” was not an MMO. Hence, students’ attention was restricted to a specific toolset for individual puzzles. (Shute & Ventura, 2013)

D. Quest Atlantis

“Quest Atlantis: Taiga Park” was a MUVE educational game designed to teach system thinking skills. The population of fish in the Taiga Park river was declining, and students were challenged to find the root problem. By travelling through time to witness cause-and-effects and question NPCs, students concluded that no single cause was responsible. A solution involving multiple variables needed to be found. (Shute, Masduki, & Donmez, 2010)
Researchers proposed, but did not implement, a stealth assessment system to analyze students’ system thinking for “Quest Atlantis”. In their proposal, they renamed the “task model” of ECD to “action model.” This choice represented their belief that an avatar’s actions can be associated with indicators that update the evidence model. In particular, they suggested a causal diagram tool that students could use to graph their beliefs of the system. Machine learning techniques could be used to analyze a student’s diagram, and offer feedback for the student and teacher. (Shute, Masduki, & Donmez, 2010)

III. EXPERIMENT CENTERED DESIGN (XCD)

“The Radix Endeavor” aims to use stealth assessment and machine learning to capture and display students’ knowledge for various audiences including teachers, students, and researchers. A number of quests in “The Radix Endeavor” have already been prototyped with a variety of variables and contexts. In order to use machine learning techniques, a standard task model must be structured to accept input from players and provide output to the evidence model. This thesis presents a modification of the ECD framework called “Experiment Centered Design” that templates quests developed for “The Radix Endeavor.” Recording quests and applying machine learning techniques requires deconstructing every quest into a shared, standard format.

A. Structure of a Quest

Defining the common quest elements is the first step to finding a uniform quest structure. Every quest must have triggers and mechanisms that allow the player to start, work on, and complete tasks.

1. Prompt

First, every quest has a prompt. The prompt initiates the quest. The primary purpose of the prompt is to inform a player what the victory condition of the quest is. The prompt often contains instructions on where and how to approach the quest. All of this information is presented in a dialogue that matches the diction and style of the story. This dialogue can be
delivered through various sources, including non-playable characters, magical objects, or simple popup directions.

2. Experimentation

Second, every quest requires experimentation. The goal of quests in “The Radix Endeavor” is to encourage scientific inquiry. Part of achieving this goal is showing that experimentation is a useful skill that is applicable to a variety of situations. In the world of “Radix,” players utilize specific skills for specific experiments, but learn the pattern of conducting steps of the scientific method: form a hypothesis, conduct a test, and analyze results.

i. Hypothesis

A hypothesis is an assumption that attempts to explain a particular phenomenon. A hypothesis is formed from one’s knowledge of the domain, which may be empty or expansive. This knowledge is essentially the student model, an unknowable state of comprehension teachers wish to understand.

ii. Test

A test is an experiment done to support or refute the hypothesis. The set of variables used in the experiment are called factors. These factors are acted upon by an operator. An experiment results in a set of observations visible to the user. In summary, an experiment is action taken on a set of factors that produces observations.

iii. Observations

The observations produced from an experiment may support, contradict, or offer no information about the hypothesis. Analyzing these observations requires separating which observations are conclusive. These conclusive observations build on the knowledge of the domain and allow one to affirm or adjust one’s hypothesis. These observations may also be used
as factors in future experiments. The cycle of hypothesizing, testing, and analyzing continues until the truth is found.

iv. Capturing the Scientific Method

Educational games that allow users to conduct experiments vary in how they present the steps of the scientific method to the player. Many presentations explicitly reveal these steps. For example, some games require players to input text describing the reasoning behind their hypothesis before they can conduct the experiment. Other games associate a particular location as a kind of “headquarters” for conducting experiments. Still other games provide players with charts of the relevant observations after an experiment. All of these options lessen the immersive experience by bringing the scientific method to the forefront of the player’s attention.

“The Radix Endeavor” supports an immersive experience by allowing users to conduct experiments anywhere in the world unimpeded by questionnaires and read-outs. The quest prototypes provided for “The Radix Endeavor” are built around tools that players can carry in their inventory to any part of the game world. Each of these tools has different operations, which include measuring, probing, and creating objects in the environment. In short, these tools allow users to conduct experiments according to the scientific method. “Radix” attempts to use a player’s actions and experiments to build the student model for that player.

3. Solution

Third, every quest has a solution. A solution is a condition that marks the completion of a quest. Solutions may be a set of predetermined values. A simple example is the binary set “true” or “false.” Solutions can also be open responses with limited constraints, such as choosing a number within a range. It is important to note that a quest may have multiple solutions, and solutions may be correct or incorrect. Players attempt to complete a quest by submitting responses that match a solution. Responses that trigger a solution are valid responses. Responses that do not match a solution and therefore do not complete the quest are considered invalid responses. These valid and invalid responses offer insight into a student’s understanding.
As an overly simplified example, imagine the question “What is the sum of adding one plus one?” This question can be solved by entering any numerical value, but the correct response is the number “2”. Any other numeral, like “11,” is an incorrect response. An invalid response is the word “two,” because it is not a numeral and cannot be accepted. Solutions in “The Radix Endeavor” are more complex and can require the manipulation of the environmental or the fulfillment of multiple conditions in order to submit a response. Yet every response is either correct, incorrect, or invalid.

4. Learning Concepts

A majority of quests are intended to teach students particular educational subjects. A learning concept is knowledge or behavior being taught or exercised in such a quest. These concepts may be correct, unusual, misguided, or simply interesting patterns of thinking or acting. A quest may also have more than one learning concept associated with it.

A valid response that matches a correct solution to a quest implies that a student understands the educational content associated with that quest. A learning objective is knowledge or behavior that is correct. In the example above, the learning concepts focus on addition. The correct response, “2,” suggests that a student has met the learning objective of understanding addition.

Quests are designed for students to ultimately succeed. However, students are expected to err while experimenting. A misconception is an error in judgment about an educational subject. Invalid responses, or valid responses that match an incorrect solution, may reveal misconceptions that the student has about the quest and its learning concepts. In the example above, the incorrect response “11” reveals a misconception about the addition operator. The invalid response “two” reveals a misconception that only numerical responses are acceptable.

5. Feedback

Finally, every quest must provide feedback for responses. This feedback has two primary audiences: players and educators. The different audiences require different feedback.
i. Player Feedback

Players desire information that advances their progress in the game. In order to maintain an immersive environment, this information must also be provided in thematically appropriate contexts. These requirements imply that players need quest feedback to be given after reaching learning objectives or falling into misconceptions. The feedback given to students may reveal these concepts in an explicit summary, or may offer gradual clues on how to proceed.

ii. Educator Feedback

Educators desire information that summarizes the learning objectives and misconceptions uncovered by students. Teachers use this information to guide students in-game and in the classroom. Educational researchers appreciate the ability to study particular patterns among students’ quest habits. For these reasons, educators require visible, easy to interpret feedback that summarizes students’ submissions after they attempt quests. In short, students need immediate feedback while experimenting in a quest, while educators need a timeline of student actions that they can review quickly.

6. Information Flow

In summary, the core elements of a “Radix” quest are prompts, experiments, solutions, and feedback. All of these elements pass information between the game system and the user. However, prompts and feedback send information in one direction from the system to the user. Experiments and their solutions offer a dialogue between the player and the system. Assessing this dialogue offers insight into the knowledge, intent, and patterns of the player.

B. Object Models

One way to represent the structure of a quest is through an object model. An object model is a diagram that maps entities and their relationships between one another. Refer to Figure 3 below. In the figure, every word surrounded by a box represents a set. An underlined word
means the set is an abstract set. Every arrow represents a relationship between sets. Arrows with open heads define a “subset” relationship. Arrows that share an open head separate two or more disjoint subsets. Arrows with filled heads define multiplicity relationships. The direction and name of the relationship defines the relation between the sets. For example, an arrow from set S to set T with the name “owns” specifies that a set S owns a set T and that a set T is owned by a set S. An arrow from set T to set S would imply the opposite, that a set T owns a set S. Multiplicities specify how many sets map to another set. The multiplicity symbols represent at most none (*), at most one (?), at least one (+), and exactly one (!). If no symbol is specified, the relationship is implied to mean at most none (*). An “attribute” relationship maps at most none (*) of set S to exactly one (!) of set T. (Jackson, 2012)

![Diagram of sets and relationships](image)

**Figure 3:** Representation of sets and relationships in an object model

Figure 4 below provides a simple example of an object model for college registration. (Jackson, 2012) The student body is partitioned into visiting students and regular students. Regular students have a degree attribute, meaning every regular student enrolls in exactly one degree program. Every student must register in at least one class. Faculty includes two subsets: advisors and teachers. Advisors may advise any number of students, but every student must have exactly one advisor. Teachers may teach at most one class, but every class must have at least one teacher.
An instance of an object model is a structure that follows the object model template with specific entities. Figure 5 below is one instance of the object model provided in Figure 4. Alice is a regular student registered in calculus and enrolled for a mathematics degree. Bob is a visiting student registered in biology. Mr. Beta is an advisor for Alice. Mr. Gamma and Mr. Delta are teachers of calculus. Mr. Epsilon teaches biology and advises Ben. Finally, Mr. Alpha is a faculty member who is neither a teacher nor an advisor. All of the constraints of the object model from Figure 4 are followed. Figure 5 represents only one instance, but there are an infinite number of instances that can be created from an object model.

Databases are essential for recording and storing the multitude of information in digital games, including information about players, quests, and the environment. Object models are useful for structuring database schemas. Database tables store objects and relationships that map objects one-to-one, one-to-many, and many-to-many. The sets and multiplicities of relationships in an object model define the tables of a relational database. Instances of the object model fill
entries in these database tables. Figure 6 shows the relational database tables for Figure 4 as well as the entities and relationships from the Figure 5 instance.

<table>
<thead>
<tr>
<th>ENTITIES</th>
<th>FACULTY</th>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>MR. ALPHA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>MR. BETA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>MR. GAMMA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>MR. DELTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>MR. EPSILON</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUDENTS</th>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ALICE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BOB</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASSES</th>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CALCULUS</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BIOLOGY</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEGREES</th>
<th>ID</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MATHEMATICS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RELATIONSHIPS</th>
<th>ADVISES</th>
<th>ID</th>
<th>TEACHER ID</th>
<th>STUDENT ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEACHES</th>
<th>ID</th>
<th>TEACHER ID</th>
<th>CLASS ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENROLLMENTS</th>
<th>ID</th>
<th>STUDENT ID</th>
<th>DEGREE ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REGISTRATIONS</th>
<th>ID</th>
<th>STUDENT ID</th>
<th>CLASS ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: College registration database tables

C. Quest Template Object Model

By abstracting the structure of quests, the quest elements and their relationships can be captured in an object model. Figure 7 is an object model that connects the users, quests, experiments, and educational content. The sets included in Figure 7 depict elements required by all quests in “The Radix Endeavor”. (Clarke-Midura, 2012)

1. Quests

Every quest is identified by its title, as well as a narrative that establishes the fiction of the quest. A quest may belong to at most one quest line, and quest lines must include at least one quest. It is possible that quests may require a time limit, or other attributes that are not shown in Figure 7. Finally, quests may require or be related to other quests.
2. Educational Content

A domain of knowledge is divided into subdomains, and every subdomain belongs to only one domain. Completion of a quest may require knowledge of any number of subdomains. Every subdomain is included in “Radix” because of some rationale. These subdomains are likely held to particular educational standards. Finally, every subdomain has one or more learning concepts attached to it. These learning concepts may be learning objectives or misconceptions, as described previously.

3. Experiments

Quests may involve conducting any number of experiments. An experiment is a particular mapping of a test to at least one observation. As described earlier, a test is an operation on a number of factors. This operation can be performed by either the real-world player or his in-game avatar. For instance, solving a mathematics problem could be done with in-game tools or by the player’s calculator in the real world. The more operations that a player executes in-game, the more data can be captured about the player’s experimentation. The operations an avatar can perform include using tools or executing actions. A test may input and operate on any number of observation factors. However, the experiment always outputs at least one observation.

4. Users

Users accept quests. When a user conducts an experiment, the experiment is logged as an event which is marked with a timestamp. These event logs can be easily retrieved and filtered to study the activity of the user. Users may have other attributes, such as their name and level, which are not currently represented in this object model.

5. Experiment Series

An experiment series is a particular pattern of one or more experiments. Experiments, when performed individually or in a specific order, may reveal certain behaviors. Finding these
patterns in a player’s event history implies that the student exhibits that behavior. Filtering by using database queries offers one solution that requires optimizing the database to handle these search queries.

Regular expressions are a powerful tool for identifying and matching patterns. Regular expressions are text strings that define a search pattern to look for. A regular expression often includes characters, wildcards, and multiplicities. For example, the regular expression 
\n```
("^1$"|([0-9]+,)+1$)
```

matches any list of numbers that end in a one. So the pattern “3,2,1” or “1” will succeed, but the pattern “3,1,2” or “11” will not. By building a regular expression for a series of experiments, one can leverage the regex grammar to search for experiment series in players’ histories. (Just Great Software Co. Ltd., 2012)

![Figure 7: Quest template object model](image-url)
D. Centering Around the Experiment

While Figure 7 gives a broad overview of important quest elements, particular sets are more interesting to different people. For example, the game writer may be interested in the quest and user attributes, but not the educational content. Curriculum developers may be solely interested in the educational content, but not how the quests, experiments, or users are involved. This paper concerns itself with learning assessment. Therefore, Figure 8 is a simplified object model that keeps the sets vital to assessing a user's educational progress.

The central set that ties this object model together is the experiment. Quests are designed around recreating experiments. Users perform experiments. Learning concepts are revealed by series of experiments. Experiments are key to designing, playing, and learning from quests.

Because of the central role experiments have in “The Radix Endeavor,” the process of developing quests according to this object model is called “Experiment Centered Assessment.”
Design” (XCD). Experiment Centered Design is an extension of Evidence Centered Design. The task model of ECD corresponds to the quests and experiments presented to players in XCD. The evidence model of ECD corresponds to the experiment series performed by players in XCD. Finally, the student model of ECD is built from the learning concepts revealed by players who perform particular series of experiments in XCD.

1. Extending Traditional Assessment in the ECD Framework

To show that Experiment Centered Design is an extension of ECD, it is important to show that traditional ECD assessment can be completed with XCD. Without the ability to match the capabilities of the ECD framework, XCD would be unable to incorporate basic assessment techniques, like questionnaires, that provide a default method for capturing student learning. First, one must show the ability for XCD to handle closed- or open-response questions. Furthermore, one should be able to capture the same data with XCD that was captured with ECD in prior work. Finally, one must show areas where XCD extends beyond the traditional ECD framework to capture unique data from open world experimentation.

i. Traditional Closed Response

First, XCD is capable of capturing closed-response questions. Consider the following scenario. Question 1 on a digital test is “All mammals have hair or fur. True or False?” This question is in the domain of biology and the subdomain of zoology. Alice and Bob take the test at 1:00 PM. Alice answers “True” and receives a green check mark next to her response indicating that she is correct. Bob answers “False” and receives a red “x” mark next to his response indicating that he is incorrect. The process of answering this multiple choice question is no different than conducting an experiment. Refer to Figure 9 below.
The quest is answering the question “Do all mammals have hair/fur?” The title of the quest is “Question 1” and its subdomain is zoology. The quest has two tests. One test is to answer “True” and the other test is to answer “False.” To perform the test, there is a multiple-choice tool that allows players to click and select their answer: “True” or “False.” Experiments map tests to outcomes. Experiment one maps the test of answering “True” to the feedback that the answer is correct (a green check mark). Performing experiment one signals that a learning objective has been met. Experiment two maps the test of answering “False” to the feedback that the answer is incorrect (a red “x”), and signals that the player has a misconception about mammal zoology.

In the above example, Alice performed the experiment stating “All mammals have hair/fur is true” and Bob performed the experiment stating “All mammals have hair/fur is false.” Each player received the appropriate feedback and their events match a series of experiments. A learning objective of understanding mammalian zoology is met when experiment one is performed. Alice performed experiment one, so she meets this learning objective. A
misconception about mammalian zoology occurs when experiment two is performed, and Bob’s experimentation history shows that he prescribes to this misconception.

ii. Traditional Open Response

An open-response question could be captured using XCD in a similar fashion. Consider the following scenario. On a digital test, question two asks “Name a barnyard member of the aves class,” and question three asks “Name a barnyard member of the mammalian class.” Both of these questions also belong in the subdomain of zoology, and Alice and Bob each reach the question at 1:01 PM. Alice receives question two and writes in “Chicken.” Bob receives question three and also writes in “Chicken.” Both students submit their answers. The system is preloaded with correct and incorrect solutions, and any responses not stored in the system are assumed to be incorrect. The system has seen Alice’s answer before, and knows that the answer is correct. The system has not seen Bob’s answer before, so by default it is marked incorrect and added to the system. Refer to Figure 10 below.
In this instance, Alice and Bob perform the same test but on different quests. They both use a write-in tool to submit “Chicken.” However, Alice’s experiment was already in the system since her response correctly answers question two. Bob’s experiment was not yet stored in the system or associated with a learning concept. By default, his experiment was added to the system as an incorrect response to question three. Because Bob’s experiment is new, it has the potential to reveal a learning concept. Indeed, this experiment also reveals a misconception about mammal zoology, and the system can be updated by an administrator, such as a teacher, to improve learning assessment. Now the misconception can be flagged when future responses match Bob’s response.

Note that under the rules of this system, correct responses could be mistakenly recorded as incorrect if the system does not recognize them. For example, imagine Bob had submitted an unconventional answer not preloaded in the system, such as “farmer.” If the system did not have this response, it would store it as “incorrect” by default. When an administrator reviews Bob’s answers, he could fix the system to accept “farmer” as a correct submission. This change would propagate through the system and update Bob’s student model. Like traditional ECD, XCD requires an agent to provide rules on approving or rejecting responses as correct or incorrect, and these rules may require updating for unique circumstances.

iii. Converting from Object Model to Database Storage

As stated previously, database tables can easily be created from object models and their instances. Figure 11 shows the database tables that correlate to the object model instances of Figures 9 and 10. The entities are the “quests,” “subdomains,” “learning concepts,” “experiment series,” “experiments,” “tests,” “operators,” “observations,” “users,” and “events” tables. The other tables define one-to-many or many-to-many relationships between the entities. This data structure offers succinct storage of the properties of the quest as well as the events performed by the users. All of these entities and their relationships can be quickly added into the database schema, either ahead of time or while a user is playing the game. Data from these quests can be retrieved quickly by performing database queries that filter for the desired information.
2. Experiment Centered Design and Related Work

XCD is capable of structuring traditional test questions. However, XCD could also be used in place of Evidence Centered Design in prior work. The click-tracking in the “River City Project” is a series of experiments that test whether interacting with an object is meaningful. The tools used in “Newton’s Playground” help players run experiments to succeed at various
challenges. By testing different “present-day” conditions in “Quest Atlantis,” players experiment to see different results in the future. All of these games are capable of adapting Experiment Centered Design since these games rely on players experimenting with their environment in order to learn and play the game.

3. Unique Features of Experiment Centered Design

Experiment Centered Design, like Evidence Centered Design, is used as a framework to guide development. While XCD can be applied to previous ECD endeavors, XCD is intended to be used during the development of quests and quest lines. In this way, Experiment Centered Design has a few unique properties.

i. Parallel Development of Experiments

First and foremost, XCD allows different designers multiple affordances to prototype experiments for educational study. Game designers may start with operations they want in-game players to perform. Curriculum designers may start with what experimental observations would reveal learning concepts. Different designers can iterate separately or work together to formulate engaging and informative experiments.

ii. Experiment Series Capture Students’ Work

Second, a series of experiments can uncover knowledge about a player’s thought process that individual responses fail to capture. Players are encouraged to perform multiple experiments in order to learn about the world and solve complex problems. The experiments and the order in which they are performed might follow patterns that reveal misconceptions or learning objectives.
iii. Randomness in Experiments

Third, experiments offer variability unexplored in traditional task models. The world of “Radix” is a complex system of objects, environments, and characters that players are encouraged to interact with. Players can experiment in the world and input consistent factors, but randomness in the system causes variable outputs. Analyzing how a student adapts to these outcomes with more experimentation offers rich insights into their strategy and learning.

IV. RESULTS

By applying the benefits of Experiment Centered Design to “The Radix Endeavor,” researchers were able to capture learning concepts in quests and experiments that were developed by independent game designers. The following examples show the object model instances and learning assessment built around experiments.

A. Quest Dialog

First, quest dialogs can be framed as experiments. Imagine an introductory quest where students are prompted to provide their gender and name. Refer to Figure 11.

Figure 11: Quest to use closed- and open-response in dialog
Asking for the player to submit their gender is no different than answering a closed-response question exemplified in Figure 9. Asking for the player to submit their name is no different than answering an open-response question exemplified in Figure 10. Figures 12 and 13 show the object model instances for completing these introductory quests. While these questions appear trivial, they test the player’s knowledge of game affordances such as radio buttons and input boxes. Later dialog boxes could present questionnaires where students submit answers to closed-or open-response questions on other educational material.

Figure 12: Introductory quest learns gender of players and introduces the radio button affordance
B. Volume and Surface Area

A quest line was developed for students to complete cost optimization problems to learn about volume and surface area. In game, players are provided with a tool to construct rectangular prisms. A non-playable character, an animal breeder, needs to travel with his animals in boxes. The first quest asks players to create a box that can fit a “tropical chicken” ten units wide, twelve units long, and fifteen units tall. If the player submits a box that is too small, the player must try again. If the player submits a box that is greater than or equal to the specified dimensions, the box is accepted.

This quest offers an interesting look into how different developers can utilize Experiment Centered Design to assess a player’s progress. Assume there are two developers: a game designer who is interested in players’ general game progress and a researcher who is interested in closely capturing student actions. Figure 14 shows the flow of the quest for a player. In the first two panels, the player approaches the breeder and accepts the quest. In the second, third, and fourth panels, the player uses his tools to create a box and add it to his inventory. In the last two panels, the player gives the box to the breeder and the breeder accepts it. In this interaction, the game designer focuses on the act of submitting a box as the experiment. The researcher focuses on the act of creating boxes as the experiment.
Figure 14: Player completing “Chicken Box” quest.
1. Quest Object Models

Assume there are two players: Alice and Bob. Alice creates a box that is $10 \times 12 \times 15$ units. She submits the box and the breeder is happy to accept it. Bob creates a box that is $1 \times 12 \times 15$ units. He submits the box, but the breeder rejects it. Bob creates another box that is $11 \times 22 \times 16$ units and the breeder is happy to accept it.

i. Game Designer Object Model

The game designer treats the act of submitting a box as the experiment. He discretizes the set of all possible experiments into three categories: submitting a box that is optimal, too small, or too big. When a box has dimensions that match $10 \times 12 \times 15$ units, the box is an exact match and the vendor accepts it. When any of the dimensions of the box are too small, the vendor rejects it. Finally, if the box is bigger than the size of the chicken, the vendor accepts it. In essence, the game designer treats the quest like a multiple choice problem. Refer to Figure 15 below.
The quest is titled “Chicken Box” and belongs to the subdomain of math that teaches volume and surface area. There are two learning objectives. One learning objective looks for a box matching the optimal solution, while the other checks whether the box is acceptable but not optimal. The only misconception occurs if any of the box’s dimensions are too small. For the game designer, experiments that test for these learning objectives are sufficient for understanding the player’s progress. Students input boxes with particular dimensions through the “box submission” dialog with the breeder. Experiment one occurs when a box with optimal dimensions is submitted. Experiment two occurs when any of the box dimensions are too small. Experiment three occurs if the box fits the chicken, but any of the dimensions are too big. Feedback for experiments one and three is that the breeder accepts the box. Feedback for
experiment two is that the breeder rejects the box. The experiments designed by the game designer capture whether students are submitting boxes that are too big, too small, or just right.

In the above instance, Alice performs experiment one, and the system recognizes that she submits an optimal solution. Bob performs experiment two, and the system flags him as having a misconception. Bob then performs experiment three, and the system records that Bob submitted an acceptable response. The system is knowledgeable of the number of submissions that Alice and Bob required to pass the quest and can differentiate optimal and suboptimal answers.

ii. Researcher Object Model

The researcher is interested in the same learning objectives. However, the researcher wants to capture more specific information about the boxes created by players. To do so, he wants to record the dimensions of every box ever created. Every box maps to one of the learning objectives. In essence, the researcher treats the quest like an open-response problem. Refer to Figure 16 below.
The quest attributes and learning objectives are identical in this object model. However, the information about box dimensions is valuable to the researcher. The box maker tool allows students to input a length, width, and height value and outputs a box with those dimensions. The researcher presumes a majority of players will submit the optimal solution. Hence, the system is preloaded with experiment one, a box made with dimensions 10 x 12 x 15 units.

In the above instance, Alice performs experiment one. The system recognizes that Alice has created a box that fits the optimal solution. Bob performs an experiment that has not been seen by the system yet. The system analyzes Bob’s creation, recognizes it as being incapable of fitting the chicken, and records it as a misconception. Bob performs another experiment that has not been seen by the system. The system evaluates this creation and records it as an acceptable box. The system is now trained with Bob’s experiments. By associating more experiments with
learning concepts, the researcher can tag other interesting phenomena, such as irregular or dyslexic submissions.

iii. Merging Object Models into Database Storage

As previously stated, one of the benefits of XCD is that designers can work independently to iterate and evaluate quests. In the example above, the system can be loaded with both sets of experiments. When creating an experiment series to flag learning concepts, the game designer and researcher filter by the experiments that meet their particular needs. Refer to Figure 17 below.
Figure 17: Database tables that combine the object model instances of the game designer and researcher for the volume and surface area quest.

Figure 17 stores the information from the object model instances provided in Figure 15 and 16. Some of the information in these figures is redundant, such as quest title and user names, and the database only needs to store this information once. The database also stores the information about the creation and submission of boxes. When querying the database, the game designer
filters entities that relate to the “box submission” operator while the researcher filters entities that relate to the “box creation” operator. Because each of these designers followed the XCD framework, all of their information is captured and stored in a routine way that can be easily retrieved and manipulated.

2. Digital Prototype

i. Experiment Emulation

A prototype of the “Chicken Box” quest was developed to demonstrate the storage and retrieval of quest data in the XCD framework. This prototype distilled the interaction to manipulating the tools used in the quest. Figure 18 is a storyboard made from screenshots of interacting with the box creation and submission tools available in the prototype.

Figure 18: Creation and submission of an acceptable box
Panels one through three show the user creating a box by entering the dimensions into the tool and obtaining an artifact in the game world. Panels four through six represent the user submitting the box to the merchant and the merchant accepting the box, causing it to be removed from view.

**ii. Data Flow and Back-end Storage**

The prototype uses Django to emulate passing data from a user-facing web site to a back-end database. Figure 19 shows the data flow of performing an experiment. The user performs the experiment, in this case by interacting with the box creation or submission tools. This data is passed to Django, which can manipulate the data and update the database. This back-end can then determine how to respond, such as providing feedback to the user. (Django Software Foundation, 2013)

![Figure 19: Data flow for recording and responding to experiments](image)

The database for the prototype is a SQLite 3 database, which offers a lightweight solution for storing the tables necessary for XCD in this prototype. Furthermore, Django offers
administration tools to easily edit and view the contents of the database. Figure 20 below shows the contents of the database that were dynamically filled as player Alice created and submitted a box with dimensions 10 x 12 x 15 units and player Bob created and submitted a box with dimensions 1 x 12 x 15 units and 11 x 22 x 16 units. All of the database table entries are sorted in chronological order from newest to oldest.

Because this data was populated dynamically, the very first event that was logged, the creation of a box with dimensions 10 x 12 x 15 units, required creating a few entries. First, the operation “makeBox” was added to the operations table, then the test “makeBox” with no input observations was added to the test table. The prototype ignores the input observation when creating a box because this data is stored as the experiment’s output. The box maker tool always outputs a box with the height, length, and width that was provided as input. So storing this data as both an input to the test and output to the experiment is redundant and unnecessary. The observation that a box was created with dimensions 10 x 12 x 15 units is recorded as an observation. Next, the complete experiment of making and outputting the box is recorded. Finally, an event is logged that Alice performed the experiment that made a box with dimensions 10 x12 x 15 units.

Submitting the box is a different experiment. When Alice submits the box, a new operation “submitBox” is added to the operations table. The input, a box with dimensions 10 x 12 x 15 units, is added to the observations table. An entry is added to the test table that defines submitting a box of size 10 x 12 x 15 units. The system recognizes that this submission is optimal, and an “optimal” experiment is added the database. Finally, the event log is updated with an entry that shows Alice performed the optimal box submission experiment. The rest of the events add new observations, tests, and experiments, but they reuse the operations of making and submitting boxes.
3. Generalization

Because MMORPGs are intended to attract a large number of players, quests often include small variations to give players similar yet unique gaming experiences. In an educational game, randomizing small differences in quests and problems is also important so players cannot cheat and obtain hard-coded answers from other players. The requirements to pass the “Chicken Box” quest could be easily adjusted by differing the length, width, or height of the animal. Hence, one
player may need to create a box for a chicken of size 10 x 12 x 15 units while another player may need to create a box for a chicken of size 9 x 13 x 15 units.

In order to capture the variations of the quest in the object model, additional sets must be added or adjusted to reflect the data being captured. However, these adjustments differ depending on how the data is being captured. In Figure 15, the game designer defines experiments that map submissions to acceptable or unacceptable responses from the animal breeder. In this case, a quest that requires a different sized chicken requires changing the input to tests that define an exact, small, and big submission. Refer to Figure 21 below. For the game designer, both of these quests share some properties like the title and subdomain. But each quest involves different experiments.

![Object model instance shares learning objectives but differs experiments](image)

The researcher would also need to update his object model instance from Figure 16. However, his object model uses learning concepts to flag for specific box creations. For this reason, the researcher could vary quests by using more specific learning concepts. Figure 22 shows an object model instance that shares experiments but differentiates the learning concepts.
Again, both of these object models could be combined by borrowing or sharing entities and relationships in the database. In addition, more design constraints may force choosing a single object model instance pattern to follow.

C. Mendelian Inheritance

A quest was developed for students to recreate Gregory Mendel’s breeding experiments in order to learn genetics. In the game, players come across a field of flowers with opposite traits A and B. The observable trait may be color, smell, or size, but the trait differs between players to prohibit students from giving the solution to other players. A local merchant agrees to pay the player if they can deliver the flower with the dominant trait. The player has no tool to determine the genotypes of these flowers, but he does have a tool that allows his avatar to cross breed two flowers and grow their children in a garden. When two flowers are crossed, however, they are destroyed in the process.
1. Quest Object Models

i. Experiment Object Models

The system has knowledge of the flowers' genotypes and knows that there are 16 possible experiments. Figure 23 is the object model instance for the experiments of this quest. The title of the quest is “Flower Breeding,” and the biology subdomain it covers is genetics. In this instance, assume flowers with trait A carry the dominant gene. A garden with all “Type A” flowers has plants with identical phenotypes but either homogeneous dominant XX or heterogeneous Xx genotypes. Hence, a player viewing a flower of type A can view the phenotype of trait A but cannot be sure of its genotype XX or Xx. Assuming trait A is dominant, “Type Ab” flowers are always heterogeneous Xx, and “Type B” flowers are always homogeneous recessive xx. Every test uses the “cross” tool, which takes two parent flowers as inputs, crosses them, and produces a garden of children. The inputs are the flower types A, B, or Ab. The outputs of an experiment are the gardens of children, which contain either all A, all B, all Ab, mixed A and B, or mixed Ab and B flower types.
Figure 23: 16 Experiments of type A, Ab, and B flower breeding
Note that different combinations of parents can yield the same garden of children, but not every garden of children can be created from crossing two flowers. Refer to Table 1 below.

Table 1: Valid combinations of flower parents and offspring

<table>
<thead>
<tr>
<th>Parent 1</th>
<th>A</th>
<th>A &amp; B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A &amp; B</td>
<td>B</td>
</tr>
<tr>
<td>A &amp; B</td>
<td>A</td>
<td>A &amp; B</td>
<td>B</td>
</tr>
<tr>
<td>Ab</td>
<td>A</td>
<td>A &amp; B</td>
<td>B</td>
</tr>
<tr>
<td>Ab &amp; B</td>
<td>A</td>
<td>A &amp; B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>Ab</td>
<td>Ab &amp; B</td>
<td>Ab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parent 2</td>
<td>Ab</td>
<td>Ab &amp; B</td>
<td>Ab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 shows the thirty theoretical combinations of flower parents and offspring. However, every cell of Table 1 that is crossed out cannot be bred. Some combinations are forbidden by the laws of genetics. For example, crossing two type B flowers will never result in type A or Ab children, because homogeneous recessive parents will never yield children with a dominant trait. Other combinations cannot be logically deduced. For example, when crossing two type A parents, one can never be certain that any of the children are type Ab, because crossing two type A parents means any of the following genotype crosses could have occurred: XX with XX, XX with Xx, or Xx with Xx. These crosses could yield children with any genotype. When crossing A and B parents, however, any children exhibiting trait A are guaranteed to be heterogeneous type Ab, because crossing a flower with genotypes XX or Xx with a flower of genotype xx can only yield flowers with genotypes Xx or xx.

Both Figure 23 and Table 1 highlight particular sets. Each of these sets represent an experiment when two parents with trait A are crossed and a child of trait B is produced. This phenomenon is the only proof that trait A is dominant. Only an Xx flower (type Ab), which expresses the dominant gene, can be crossed with itself to produce the recessive xx flower (type B). A player must perform one of these experiments as well as recognize this phenomenon in order to correctly complete the quest.
While Figure 23 is a verbose representation of each experiment, it contains multiple, redundant references to information. Figure 23 does not capture the elegance of using only three observations and an operator with a maximum of two inputs/outputs that produces a myriad of experiments. Figure 24 eliminates the redundant sets in Figure 23 while still mapping every relationship in the quest.

Figure 24: Concise view of type A, Ab, and B flower breeding

ii. Learning Concept Object Models

Figures 23 and 24 exhaust the experiment relationships, but fail to include the sets of learning concepts and experiment series. Figure 25 shows a number of possible learning concepts and the
experiment series that reveal them. Each of the experiment series is written as a regular expression. This expression assumes that each experiment number in a player’s event history is concatenated onto a string followed by a comma. The comma acts to tokenize the different experiment events. As an example, if a student performed experiment one, then experiment five, and finally experiment seven, the result would be “1,5,7,”.

Figure 25: Learning concepts and the experiment series that reveal them for the “Flower Breeding” quest

The first learning objective checks that a solution has been witnessed. As previously stated, a solution results when two flowers with trait A are crossed and yield a flower with trait B. Performing experiments two, three, nine, ten, twelve, or thirteen at any time reveals this learning objective. If a player only performs experiment two or three, he found the solution on his first cross. While not a learning objective or misconception, this phenomenon may be an interesting occurrence to flag. The player found the solution with minimal interaction with the game, which could make the player feel lucky or bored.

The last learning objective checks for a series of experiments that have a high probability of yielding the correct solution. Performing this series of events shows careful premeditation to
solve the problem. First, a student performs experiment five, crossing $A$ and $B$ flower types. This first cross yields flowers that are either heterogeneous $Xx$ (type $Ab$) or homogeneous recessive $xx$ (type $B$). The student then crosses children flowers with the same phenotype hoping to witness a solution. If the student crosses two type $B$ children, he will perform experiment seven and need to try the other cross. If the student crosses two type $Ab$ children and yields a garden with any type $B$ flowers, he has performed experiment twelve or thirteen and found a solution. If the student is unlucky and performs experiment eleven and seven, he should start over with experiment five. The regular expression checks whether a student performed experiment five and experiment twelve/thirteen. It also does not penalize a student for performing experiment seven before experiment twelve/thirteen.

Certain series of experiments can also reveal misconceptions. If a student only crosses plants of the same type for more than three experiments, then the student is likely hoping to get lucky without performing the more stable method described above. This behavior is flagged, allowing for intervention. Lastly, if a student performs an experiment that reveals the solution, but he continues to perform experiments, then it is likely he does not comprehend that he has found a solution. On the other hand, the student may simply enjoy running experiments. The system cannot differentiate between these two cases, and always reports experimentation after a solution has been found as a misconception.

### iii. Player Object Models

As an example, assume that type $A$ flowers are red in color and type $B$ flowers are white in color. A variety of players receive the quest from the merchant and come across a field of red and white flowers. Before performing any actions, the players can be certain that only red and white flowers can be bred and that one of these flower colors is a dominant trait. Players already know that they can use the “cross” tool to breed flowers. This breeding results in growing a garden of all red, all white, or a mix of flowers. Because the players cannot be certain which of the plants are heterogeneous, the players are limited to completing one of the first seven experiments shown in Figure 23.
a. Player Alice

Player Alice knows that only crossing two heterogeneous flowers will breed a flower of the opposite, recessive trait. Hence, Alice crosses a red and white flower and grows a garden of mixed flowers. She knows that one set of these children are heterogeneous and the other set of these children are homogeneous recessive, but she cannot be certain which is which. She guesses, and crosses two of the red children. Again, she receives a mixed garden. Alice now knows that red plants carry the dominant gene and white plants carry the recessive gene. She submits her answer to the merchant.

The system recorded that Alice performed experiment five followed by experiment twelve. This series of experiments matches the learning objective that a solution has been found. Alice submitted the correct answer to the merchant, and the system is more confident that Alice observed the solution rather than simply guessing. Figure 26 shows an object model instance with the sets relative to Alice’s play experience.

![Diagram showing the experiment series and observations related to Alice's play experience in the Flower Breeding quest.](image-url)
b. Player Bob

Player Bob knows that the dominant trait is more likely to breed similar children. Bob crosses a red and white flower, and he produces a garden with all white children. Bob performs the same test and produces a garden with more white flowers than red flowers. Bob is convinced that white is the dominant flower, and submits his answer to the merchant.

The system recorded that Bob performed experiment six then experiment five. The system also records Bob’s incorrect answer. Figure 27 shows an object model instance with the sets relative to Bob’s play experience.

No specific misconceptions were flagged during Bob’s experimentation, but a teacher looking at Bob’s event history could infer that Bob took the majority of white children to mean white was the dominant trait. The teacher, a researcher, or any other administrator could add a new misconception to the system that flags when a student performs experiment six or sixteen and submits the incorrect answer. The system could then catch future students who make the same error as Bob.
c. Player Charlie

Player Charlie knows that any experiment where two flowers of the same color create a child of a different color reveals that the parents carry the dominant color trait. Charlie decides to cross two red plants, which results in all red children. Charlie then crosses two white plants, which results in all white children. Charlie repeats crossing two red flowers, and receives all red children. Finally, Charlie crosses two white children and receives all white children. Charlie is stuck, guesses that the dominant color is red, and submits his answer to the merchant.

The system recorded that Charlie performed the experiments one, seven, one, and seven. This series triggers the misconception of “Trial and Error Breeding.” While Charlie submitted the correct answer, he did not trigger the learning objective of finding a solution. Hence, the system is aware that Charlie merely guessed correctly and can intervene if necessary. Figure 28 shows an object model instance with the sets relative to Charlie’s play experience.

Figure 28: Object model instance for Charlie playing Flower Breeding quest
2. Digital Prototype

i. Experiment Emulation

A prototype of the “Flower Breeding” quest was developed to demonstrate a simple visualization of updating the student model in XCD. Again, this prototype simplifies the quest experience to interacting with the breeding tool. The player clicks and drags two flowers to breed them and grow a garden. The player can then click and drag particular flowers to save for breeding in the future. Figure 29 storyboards Alice’s experience breeding flowers from Figure 26. Panels one through four show Alice crossing flowers of type A and B, which yields a garden with Ab and B flowers. Panel five shows Alice saving two copies of both the red and white flowers from her last cross. Panels six through eight show Alice crossing the type Ab flowers she saved, producing red and white flowers, and witnessing the solution to the quest.

Figure 29: Experimentation crossing A x B, then crossing Ab x B
ii. Data Flow and Back-end Storage

This prototype follows the same data flow from Figure 19. The player performs an experiment, which is submitted to the web back-end and updates the database. Figure 30 shows the database tables for the prototype. The system can use the player’s event history and machine learning techniques to search for series of experiments that reveal learning concepts. This information can be used to update the student model and provide feedback to the player, the teacher, or any other interested party.

![Figure 30: View of SQLite tables for “Flower Breeding” quest prototype using Django administrator tools](image)
iii. Adjusting Student Model From Experimentation

In this quest, every experiment offers information about the probability that a flower carries the dominant trait. Bayes’ Theorem states that the probability of an event $N$ given an event $M$ is equivalent to the probability of event $M$ given event $N$ multiplied by the probability of event $N$ divided by the probability of event $M$, or:

$$P(N|M) = \frac{P(M|N)P(N)}{P(M)} = \frac{P(M|N)P(N)}{P(M|N)P(N) + P(M|\bar{N})P(\bar{N})}$$

Let $N$ be the event that the red flowers carry the dominant trait and $M$ be the event of a particular experiment taking place. Then Bayes’ Theorem can calculate the probability that red color is the dominant trait given any experiment. Because either red or white color must be the dominant trait, the sum of the probability that red flowers carry the dominant trait and the probability that white flowers carry the dominant trait is equal to one, or:

$$P(\text{red color is dominant}) + P(\text{white color is dominant}) = P(R) + P(W) = 1.0$$

Finally, the player should start with no bias about which color is the dominant trait and must assume each color has an equal probability of being the dominant trait, or:

$$P(R) = P(W) = 0.50$$

Given these facts, the system can determine what a player should assume is the dominant trait given their history of experiments.

a. Probability that Red Color Trait is Dominant Given Experiment One Occurs

As an example, one can calculate the probability that red color is the dominant trait given that a player has only performed experiment one. The formula is:
\[ P(R|Ex 1) = \frac{P(Ex 1|R)P(R)}{P(Ex 1|R)P(R) + P(Ex 1|W)P(W)} \]

The outcome of any experiment is a garden with all red flowers, all white flowers, or a mix. The probability of each outcome depends on the number of children grown in a garden. In this prototype, eight children are yielded from crossing two parents. Experiment one states that two type A (red) flowers were crossed and yielded a garden with all type A offspring.

Given experiment one is the first experiment performed by the player, the probability that trait A is dominant is about forty-four percent. This calculation requires finding the values used in Bayes’ Theorem, which are outlined below.

1. Probability that Experiment One Occurs Given Red Color Trait is Dominant

Figure 31 shows the Punnett squares for the three possible crosses that could have occurred between two type A parents when A is the dominant trait. The probability that a single child from cross one or two is red in color is one-hundred percent, and the probability that a single child from cross three is red in color is seventy-five percent.

The probability that a garden with eight children contains all red flowers is the probability of a single flower being red raised to the power of eight. Hence, the probability of performing
experiment one is the probability of each of these crosses yielding eight red flowers weighted by
the probability of that cross occurring, or:

\[ P(Ex1|R) = P(Cross \ 1) * (1.0^8) + P(Cross \ 2) * (1.0^8) + P(Cross \ 3) * (0.75^8) \]

I. Probability of Selecting Parent Flower Genotypes

The probability of each cross occurring depends on the assumption of the player choosing a
parent with the XX or Xx genotype. The probability of each cross is:

\[ P(Cross \ 1) = P(XX \ genotype \ parent)^2 \]
\[ P(Cross \ 2) = 2 * P(XX \ genotype \ parent) * P(Xx \ genotype \ parent) \]
\[ P(Cross \ 3) = P(Xx \ genotype \ parent)^2 \]

A. Flowers Selected from a “Stable State” Garden

One option would be to assume that the flowers in the wild have reached a “stable state” after
many generations of breeding, which would result in a population that mimics the yield of cross
three from Figure 31: twenty-five percent XX genotype, fifty percent Xx genotype, and twenty-
five percent xx genotype. Given that the player chose a type A flower, he has a one-third chance
of selecting a flower with XX genotype and a two-thirds chance of selecting a flower with a Xx
genotype. So the probability of each of the crosses would be:

\[ P(Cross \ 1) = \left(\frac{1}{3}\right)^2 = \frac{1}{9} \]
\[ P(Cross \ 2) = 2 * \frac{1}{3} * \frac{2}{3} = \frac{4}{9} \]
\[ P(Cross \ 3) = \left(\frac{2}{3}\right)^2 = \frac{4}{9} \]
B. Flowers Selected with Equal Probability

The option above assumes that the player discovers a growth of wild flowers in the game in a “stable state”. However, if a player assumes the wild flowers are in a stable state, he assuming that the color of the majority of flowers is the dominant trait. Therefore, the game is unlikely to offer an arrangement of wild flowers in the “stable state.” In this case, the player could assume that he randomly chooses between flowers with an XX or Xx genotype. Now the probabilities of the crosses are:

\[
P(\text{Cross 1}) = \left( \frac{1}{2} \right)^2 = \frac{1}{4}
\]
\[
P(\text{Cross 2}) = 2 \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}
\]
\[
P(\text{Cross 3}) = \left( \frac{1}{2} \right)^2 = \frac{1}{4}
\]

Using these weights, the formula that experiment one occurred given red is the dominant trait is:

\[
P(\text{Ex 1}|R) = \frac{1}{4} (1.00^8) + \frac{1}{2} (1.00^8) + \frac{1}{4} (0.75^8) \approx 78\%
\]

2. Probability that Experiment One Occurs Given White Color Trait is Dominant

If one assumes trait B is the dominant trait, then crossing two flowers with trait A is always crossing two homogeneous recessive flowers. Crossing two homogeneous recessive parents always yields a garden of all homogeneous recessive children, all of which share the same phenotype. Hence, there is a one-hundred percent chance that crossing two red colored flowers yields a garden of all red flowers if white is the dominant trait, or:

\[
P(\text{Ex 1}|W) = 100\%
\]
3. Probability that Red Color Trait is Dominant Given Experiment One Occurs as First Experiment

Following the formula using Bayes' Theorem, the player knows the probability of experiment one given that red color is the dominant trait, the probability of experiment one given that white color is the dominant trait, the starting probability that red color is the dominant trait, and the starting probability that white color is the dominant trait. Evaluating using these values yields:

\[
P(R|Ex\ 1) = \frac{P(Ex\ 1|R)P(R)}{P(Ex\ 1|R)P(R) + P(Ex\ 1|W)P(W)} = \frac{(0.78)(0.50)}{(0.78)(0.50) + (1.0)(0.50)} \approx 44%\]

Now a player can assume that red color is the dominant trait with forty-four percent certainty, rather than an initial guess of fifty percent certainty.

b. Visualizing Student Model Updates Over Time

Table 2 provides the probabilities that an experiment will occur assuming trait A or B is dominant.
Table 2: Probabilities for an experiment being performed assuming trait A or B is the dominant trait

<table>
<thead>
<tr>
<th>THE PROBABILITY THAT CROSSING ...</th>
<th>... YIELDS ...</th>
<th>... GIVEN TRAIT A IS DOMINANT.</th>
<th>... GIVEN TRAIT B IS DOMINANT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A x A</td>
<td>All trait A children (Ex 1)</td>
<td>0.78</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Mixed children (Ex 2)</td>
<td>0.22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>All trait B children (Ex 3)</td>
<td>0.0000038</td>
<td>0</td>
</tr>
<tr>
<td>A x B</td>
<td>All trait A children (Ex 4)</td>
<td>0.50</td>
<td>0.0020</td>
</tr>
<tr>
<td></td>
<td>Mixed children (Ex 5)</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>All trait B children (Ex 6)</td>
<td>0.0020</td>
<td>0.50</td>
</tr>
<tr>
<td>B x B</td>
<td>All trait B children (Ex 7)</td>
<td>1.0</td>
<td>0.78</td>
</tr>
<tr>
<td>A x Ab</td>
<td>All trait A children (Ex 8)</td>
<td>0.55</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Mixed children (Ex 9)</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>All trait B children (Ex 10)</td>
<td>0.0000076</td>
<td>0</td>
</tr>
<tr>
<td>Ab x Ab</td>
<td>All trait A children (Ex 11)</td>
<td>0.10</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Mixed children (Ex 12)</td>
<td>0.90</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>All trait B children (Ex 13)</td>
<td>0.000015</td>
<td>0</td>
</tr>
<tr>
<td>Ab x B</td>
<td>All trait A children (Ex 14)</td>
<td>0.0039</td>
<td>0.0020</td>
</tr>
<tr>
<td></td>
<td>Mixed children (Ex 15)</td>
<td>0.99</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>All trait B children (Ex 16)</td>
<td>0.0039</td>
<td>0.50</td>
</tr>
</tbody>
</table>

As described above, these values can be used to calculate the probability that trait A or B is the dominant trait. The example above describes how a student’s assumptions should change if their first experiment was experiment one. Table 3 shows the formulas to update the probability that trait A is dominant given any experiment has been performed. The final column of the table shows an example given that the probability trait A is dominant is equal to the probability that trait B is dominant, like before the player has performed an experiment.
Table 3: Formulas for updating the probability that trait A is dominant given any of the sixteen experiments

<table>
<thead>
<tr>
<th>PROBABILITY TRAIT A IS DOMINANT GIVEN ...</th>
<th>BAYES’ THEOREM FORMULA</th>
<th>FORMULA WITH VALUES PROVIDED IN TABLE 2</th>
<th>GIVEN</th>
<th>P(R) = P(W) = 0.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>( P(Ex 1</td>
<td>R)P(R) )</td>
<td>( 0.78P(R) + (1.0)P(W) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 2</td>
<td>( P(Ex 2</td>
<td>R)P(R) )</td>
<td>( 0.22P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 3</td>
<td>( P(Ex 3</td>
<td>R)P(R) )</td>
<td>( 0.0000038P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 4</td>
<td>( P(Ex 4</td>
<td>R)P(R) )</td>
<td>( 0.50P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 5</td>
<td>( P(Ex 5</td>
<td>R)P(R) )</td>
<td>( 0.50P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 6</td>
<td>( P(Ex 6</td>
<td>R)P(R) )</td>
<td>( 0.20P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 7</td>
<td>( P(Ex 7</td>
<td>R)P(R) )</td>
<td>( 1.0P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 8</td>
<td>( P(Ex 8</td>
<td>R)P(R) )</td>
<td>( 0.55P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 9</td>
<td>( P(Ex 9</td>
<td>R)P(R) )</td>
<td>( 0.45P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 10</td>
<td>( P(Ex 10</td>
<td>R)P(R) )</td>
<td>( 0.0000076P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 11</td>
<td>( P(Ex 11</td>
<td>R)P(R) )</td>
<td>( 0.10P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 12</td>
<td>( P(Ex 12</td>
<td>R)P(R) )</td>
<td>( 0.90P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 13</td>
<td>( P(Ex 13</td>
<td>R)P(R) )</td>
<td>( 0.000015P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 14</td>
<td>( P(Ex 14</td>
<td>R)P(R) )</td>
<td>( 0.0039P(R) )</td>
<td></td>
</tr>
<tr>
<td>Experiment 15</td>
<td>( P(Ex 15</td>
<td>R)P(R) )</td>
<td>( 0.99P(R) )</td>
<td></td>
</tr>
</tbody>
</table>

The student’s assumption that trait B is dominant is equal to one minus the probability that trait A is dominant. Note how the formulas for experiments two, three, nine, ten, twelve, and thirteen reduce to one-hundred percent probability regardless of the current assumption of trait A or B being dominant. This follows logically since performing any of these experiments proves that trait A is dominant.

As an example, in Figure 26 player Alice performs experiment five by crossing type A and B parents to yield a mixed garden. Logically and statistically, she has gained no new information, since a mixed garden occurs with a high probability regardless of whether trait A or B is dominant. Until she performs another experiment, Alice can still only guess with fifty-fifty probability which flower carries the dominant trait. When Alice performs experiment twelve,
however, she can be certain that the red color is the dominant trait. The system not only captures that Alice should have the solution to this quest, but her series of experiments matches a learning objective and is flagged in the prototype.

Figure 32 is a simple visualization that shows a timeline of the certainty to which Alice should believe red color is the dominant trait. In the first panel, she assumes red has a fifty percent probability of being the dominant trait. Each experiment affects Alice’s perception, either maintaining her knowledge (after performing experiment five in panel two) or adding to it (after performing experiment twelve in panel three). Notice that the visualization also reports flags, such as when Alice found the solution after experiment twelve. This flag was thrown because her event history matched a regular expression of an experiment series that solves the quest. This visualization is an example of displaying the student model for the flower breeding quest. The graph shows the most recent understanding that Alice has of the system and flags important events Alice triggered while experimenting. At a higher level, only the flags from each quest are needed to feed a machine learning algorithm that cumulates a player’s educational and game progress.
3. Generalization

In the examples above, traits A and B were the colors red and white. In order to provide variation for different players, the traits could use different colors or even different properties of the plant, like size or smell. As long as these traits follow the rules of complete dominance, the trait characteristics are unimportant to the back-end. The database still captures players
experimenting with type A, B and Ab flowers and can flag any player's actions that match a learning concept.

If the inheritance rules change, such as following the rules of codominance or incomplete dominance, then the experiments that a player performs also change. The new observations, tests, learning concepts, and experiment series would be added using the XCD framework in order to capture and flag players' actions.

V. FUTURE WORK

Experiment Centered Design provides a framework for assessing student learning and progress by tracking experiments completed by the students. Future work with XCD has potential to utilize and build upon this framework. Utilizing this design template allows other educational game developers to capture players experimenting within their game and map these actions to learning concepts. Building upon XCD allows developers to filter and manipulate the data collected.

A. Experiment Centered Design in Future Endeavors

First, a number of other quests have been proposed for use in “The Radix Endeavor.” However, at the time of writing, a limited number of quests have moved into the prototyping stage. This paper describes the application of XCD to those quests most likely to appear in “The Radix Endeavor” in their final form. The opportunity exists to take other quests in “Radix” and apply the XCD framework. This work will be completed for each subsequent quest added to “Radix.”

Apart from “The Radix Endeavor,” a number of other educational games could make use of XCD. As previously mentioned, the educational games “River City Project,” “Newton’s Playground,” and “Quest Atlantis” could utilize Experiment Centered Design to restructure their learning assessment. Future work could frame the actions in a quest or level from one of these games to fit with XCD. This work could help compare the pros and cons of utilizing ECD, Stealth Assessment, and XCD. Such a comparison could look at the ease of adoption, storage memory requirements, and versatility to rapid iteration.
B. Classification of Experiments in Experiment Centered Design

Experiment Centered Design extends ECD by giving specific form to the task, evidence, and student models. The ability to specify additional patterns in XCD may benefit developers by providing examples to guide their work. Categorizing experiments into particular families may help developers quickly fit an experiment into a particular XCD template.

This paper has begun to recognize differences in particular sets of experiments. One example is the application of XCD to closed- and open-response questions. Experiments that resemble open-response questions dynamically populate the database of tests performed by players. Experiments that resemble multiple-choice questions, however, can exhaust the list of tests performed by players and prepopulate the database.

The volume and surface area quest exposed another set of different experiments. The experiments developed by the game designer focused on the player submitting information for review. Players submitted a box with a particular length, width, and height to be judged by the breeder. The experiments developed by the researcher focused on the player submitting information to create an artifact. The player input a length, width, and height to create a box. The in-game questionnaires resemble the former, where students input selections and text to be interpreted by the system. The flower breeding quest resembles the latter, where students input flowers to create a garden of more flowers.

With more examples of quests utilizing XCD, more patterns and classifications are likely to arise. By exploring these different classifications, the ability to apply XCD to new and different experiments will become easier and more efficient.

C. Social Experiments in Experiment Centered Design

One set of quests that has yet to be prototyped require multiple players interacting with each other as well as the environment. These “multiplayer” quests may require the players to perform series of experiments dependent on each other. Because the players share the set of experiments needed to solve the quest, XCD must analyze the union of the players’ actions to interpret
D. Feedback Portals

Experiment Centered Design forms a model of a student based upon learning concepts. How this model is utilized and displayed depends on the audience. Students benefit from feedback that can immediately coach and correct their behavior. Teachers benefit from the aggregation of student performances so they can teach to students’ weaknesses. Researchers benefit from watching trends so they can identify patterns in students’ actions.

In “The Radix Endeavor,” opportunity exists to experiment with the feedback systems for the students, teachers, and researchers. For students, the feedback system should highlight a student’s successes while encouraging a student to think deeper about his misconceptions. This system could take the form of an NPC assistant that follows the player. This system could also take the form of a graphical user interface that lists achievements, or this system could be woven into the dialog with quest givers. All of these systems have pros and cons that balance immersing the student in the fiction of the game and urging the student to consider the real-world applications beyond the game.

For teachers, the feedback system must be a glanceable UI that brings students’ misconceptions into focus. The game is a supplement to the teacher’s curriculum, and treats quests with the same gravity as homework. For this reason, the teacher is more interested in the common struggles holding back the majority of his class instead of the individual actions of each player. In addition, a user interface that mimics tools that teachers are already familiar with, like grade book software, may help teachers comprehend the data more easily.

For researchers, the feedback system should be able to analyze trends across the entire population of players. This feedback system may require filtering demographics or searching for specific patterns. This feedback system may also require researchers to dynamically add flags to capture future phenomena. Overall, the user interface must empower researchers to identify the strengths and weaknesses of the project through the performance of the students.
All of these interfaces can utilize students’ learning concepts as input. Future work is needed to filter and display the information in the most appeasing manner to a particular audience. Iterating a design based on the audience’s needs may require significant consideration and work.

VI. CONCLUSION

This paper described Experiment Centered Design (XCD), a framework for learning assessment that extends Evidence Centered Design. The paper provided an object model that templates a game’s design around experiments. “The Radix Endeavor” is a massively multiplayer online educational game that utilized XCD to capture students learning concepts through gameplay. Future work hopes to further refine the classification of experiments captured in XCD and display the data captured in meaningful interfaces for a variety of audiences.

REFERENCES


