ABSTRACT

DOCUMENTING DESIGN RATIONALE TO PROMOTE LEARNING

by John Malloy

Design rationale documents the decision-making and reasoning that goes on when faced with a design problem. For students, this thought process often takes place mentally and with little documentation. This raises the interesting question of whether or not requiring documentation of design rationale would lead to improved decision-making and learning. In order to address this question, SEURAT_Edu was developed as a web-based rationale documentation and inferencing application. By requiring the use of SEURAT_Edu within course assignments at Miami University, meaningful data was collected and analyzed to determine the effectiveness of rationale documentation in student learning.
DOCUMENTING DESIGN RATIONALE TO PROMOTE LEARNING

A Thesis

Submitted to the

Faculty of Miami University

In partial fulfillment of

The requirements for the degree of

Master of Computer Science

Department of Computer Science and Software Engineering

by

John W. Malloy

Miami University

Oxford, Ohio

2014

Advisor____________________

Janet Burge

Reader____________________

Bo Brinkman

Reader____________________

Jason Abbitt
Table of Contents

1 Introduction 1
   1.1 Research Questions 1
   1.2 Outline 2

2 Related Work 2
   2.1 Cognitive Development 2
   2.2 Creativity 5
   2.3 Critical Thinking 5
   2.4 Use of Rationale in Education (Argumentation-Based Rationale) 6

3 SEURAT_Edu Approach 6
   3.1 Rationale Types 6
   3.2 Create Assignment 7
   3.3 Complete Assignments 8
   3.4 View Student Turnins 8
   3.5 Rationale Inferencing 8

4 SEURAT_Edu Architecture 8
   4.1 Overview 8
   4.2 Apache Server 9
   4.3 SEURAT_Edu 10
   4.4 SEURAT_Edu Database 10
      4.4.1 Tables 10
   4.5 Niihka Learning Management System (LMS) 10
      4.5.1 LTI Standards 10
      4.5.2 Problems Encountered 11

5 SEURAT_Edu Implementation 11
   5.1 Adding SEURAT_Edu to Niihka Site 12
   5.2 Home Screen 12
   5.3 Roster 13
   5.4 Groups 15
   5.5 Assignment Creation 16
   5.6 Rationale Tree 18
5.7  Rationale Editor  
5.7.1  Requirements  20  
5.7.2  Decisions  21  
5.7.3  Alternatives  22  
5.7.4  Arguments  23  
5.7.5  Claims  25  
5.7.6  Assumptions  26  
5.7.7  Help Panel  27  
5.8  Status Panel  27  
5.8.1  Inference Types  28  

6  Experiments  
6.1  Design  29  
6.2  CSE274 – Data Structures  29  
   6.2.1  Demonstration  31  
   6.2.2  Analysis  31  
   6.2.3  Survey  33  
6.3  CSE201 and CSE322  34  
   6.3.1  Demonstration  36  
   6.3.2  CSE201 Analysis  37  
   6.3.3  CSE322 Analysis  39  
   6.3.4  Analysis Comparison  41  
   6.3.5  CSE201 Survey  44  
   6.3.6  CSE322 Survey  46  
   6.3.7  Survey Comparison  47  

7  Future Work  
7.1  SEURAT_Edu Development  50  
    7.1.1  Roster  50  
    7.1.2  Group Assignments  50  
7.2  Experiments  50  
    7.2.1  Comparison Experiment  50  
    7.2.2  Nontechnical Experiment  51  
    7.2.3  Extended Experiment  51  

7
List of Tables

Table 1 Bloom’s Taxonomy, Reflection and Rationale, and Perry’s Scale (Burge and Brinkman 2010). 4
Table 2 Rationale inference types 28
Table 3 Assignment Comparison 44
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SEURAT_Edu Architecture Overview</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Instructor home screen</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Assignment deletion prompt</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Assignment turnins page</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Roster creation page, before</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Roster creation page, after</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Group creation page, before</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Group creation page, after</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>Instructor assignment creation page</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>&quot;Create Student Version&quot; warning prompt</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Rationale Tree</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>Initial rationale editor</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>Creating a requirement</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>Creating a decision</td>
<td>21</td>
</tr>
<tr>
<td>15</td>
<td>Creating an alternative</td>
<td>22</td>
</tr>
<tr>
<td>16</td>
<td>Creating an argument</td>
<td>23</td>
</tr>
<tr>
<td>17</td>
<td>Creating a claim</td>
<td>25</td>
</tr>
<tr>
<td>18</td>
<td>Creating an assumption</td>
<td>26</td>
</tr>
<tr>
<td>19</td>
<td>Rationale editor help panel</td>
<td>27</td>
</tr>
<tr>
<td>20</td>
<td>Status Panel</td>
<td>27</td>
</tr>
<tr>
<td>21</td>
<td>CSE274 Student Distribution</td>
<td>30</td>
</tr>
<tr>
<td>22</td>
<td>CSE274 starting rationale</td>
<td>30</td>
</tr>
<tr>
<td>23</td>
<td>Adopted Solutions</td>
<td>31</td>
</tr>
<tr>
<td>24</td>
<td>CSE274 Arguments Changed</td>
<td>32</td>
</tr>
<tr>
<td>25</td>
<td>CSE274 Statuses</td>
<td>33</td>
</tr>
<tr>
<td>26</td>
<td>CSE201 Student Distribution</td>
<td>34</td>
</tr>
<tr>
<td>27</td>
<td>CSE274 starting rationale</td>
<td>34</td>
</tr>
<tr>
<td>28</td>
<td>CSE201 Starting Rationale</td>
<td>35</td>
</tr>
<tr>
<td>29</td>
<td>Assignment 2 Demonstration</td>
<td>37</td>
</tr>
<tr>
<td>30</td>
<td>CSE201 Decision 1 Alts</td>
<td>37</td>
</tr>
<tr>
<td>31</td>
<td>CSE201 Decision 1 Selected Alts</td>
<td>38</td>
</tr>
<tr>
<td>32</td>
<td>CSE201 Decision 2 Selected Alts</td>
<td>39</td>
</tr>
<tr>
<td>33</td>
<td>CSE201 Decision 3 Alts</td>
<td>39</td>
</tr>
<tr>
<td>34</td>
<td>CSE322 Decision 1 Alts</td>
<td>40</td>
</tr>
<tr>
<td>35</td>
<td>CSE322 Decision 1 Selected Alts</td>
<td>40</td>
</tr>
<tr>
<td>36</td>
<td>CSE322 Decision 2 Selected Alts</td>
<td>41</td>
</tr>
<tr>
<td>37</td>
<td>CSE322 Decision 3 Alts</td>
<td>41</td>
</tr>
<tr>
<td>38</td>
<td>Decision 1 Alts Comparsion</td>
<td>42</td>
</tr>
<tr>
<td>39</td>
<td>Decision 1 Selected Alts Comparison</td>
<td>43</td>
</tr>
<tr>
<td>40</td>
<td>Decision 2 Selected Alts Comparison</td>
<td>43</td>
</tr>
<tr>
<td>41</td>
<td>CSE201 More Alternatives</td>
<td>44</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>42</td>
<td>CSE201 More Thought</td>
<td>45</td>
</tr>
<tr>
<td>43</td>
<td>CSE201 Use SEURAT_Edu Again</td>
<td>45</td>
</tr>
<tr>
<td>44</td>
<td>CSE322 More Alternatives</td>
<td>46</td>
</tr>
<tr>
<td>45</td>
<td>CSE322 More Thought</td>
<td>47</td>
</tr>
<tr>
<td>46</td>
<td>CSE322 Use SEURAT_Edu Again</td>
<td>47</td>
</tr>
<tr>
<td>47</td>
<td>More Alternatives Comparison</td>
<td>48</td>
</tr>
<tr>
<td>48</td>
<td>More Thought Comparison</td>
<td>49</td>
</tr>
<tr>
<td>49</td>
<td>Use SEURAT_Edu Again Comparison</td>
<td>49</td>
</tr>
</tbody>
</table>
Appendices

Appendix A SEURAT_Edu MySQL Tables 54
Appendix B Student Waiver 56
Appendix C Student Demographic Survey 57
Appendix D Student Post-Test Survey (CSE274) 58
Appendix E Student Post-Test Survey (Revised) 60
Acknowledgements

First and foremost I would like to thank Dr. Burge for getting me involved with this research. Without her support and guidance, none of this would have been possible. I would also like to thank Tom Barber, who was the student that helped onboard me when I first joined the team. He worked on SEURAT_Edu before me and provided help in the form of direct guidance. During development, Dr. Campbell provided support in the form of providing and setting up the web server where SEURAT_Edu is hosted. Another faculty member that was critical to this research’s success is Dr. Brinkman. He took a leap of faith in being the first instructor to let me use SEURAT_Edu in his classroom, agreeing even before the tool was in a functioning state. Finally, I would like to thank Jason Abbitt, Janet Burge, and Bo Brinkman for the time and energy they put into being on my thesis committee. Without the contributions of each of these individuals, SEURAT_Edu and the rest of this research would be nowhere near as complete as it is today.
1 Introduction

Students struggle with the act of identifying multiple solutions to a single problem. They can get locked into thinking that the most familiar solution is the best solution, which is not always the case. Though the obvious issue is that viable solutions for their particular problems may never be analyzed, a potentially more serious issue is that the students don’t learn how to consider multiple solutions to problems in general. The hope is that by having students document their design rationale, which consists of the decisions and reasoning that are involved when faced with design problems, the students will learn to consider multiple solutions to problems.

More generally, the problem that has been observed in the classroom is that students are not always proficient at analyzing the problems with which they are faced. Many students have trouble not only in articulating the decisions they are faced with, but also in coming up with and considering multiple alternative solutions. When faced with a decision, some students have shown a tendency to go with the solution that they have most recently talked about in class or that they believe they are the best at. The reality is that students are making decisions simply based on their own proficiency with a certain solution and the ease of implementation, instead of based on what is best for the problem at hand. Students must be challenged to consider multiple alternatives and weigh the pros and cons of each alternative before making a decision.

The biggest roadblock preventing students from documenting design rationale is typically a lack of available software for the task. SEURAT_Edu was created to fill this void as an easily accessible rationale documentation tool. Linked to Miami University’s online learning management system (LMS), Niihka, SEURAT_Edu is easy for both instructors and students to use. The tool provides an interface for documenting rationale and inferences over the input to provide immediate feedback about the rationale. By utilizing a unique assignment creation process, instructors can even provide their students with rationale in order to get them started. Through the use of SEURAT_Edu, students should be challenged to more actively consider their design.

1.1 Research Questions

This research was driven by a single main idea: “Does documenting design rationale provide meaningful benefits for student learning?” This question is very broad, though, and can be broken down into a few distinct questions.

- Is there a relationship between rationale documentation and cognitive development?
- Can rationale be used to identify gaps in student knowledge?
- Does rationale documentation positively impact the decision making process?

However, as there were no easily accessible rationale documentation tools, this led to the additional goal of creating such a piece of software because without a useful tool, running experiments would not provide any meaningful results. For this reason, the primary focus has been the development of SEURAT_Edu and the completion of several preliminary
experiments. After this, the research can be continued with a heavier focus on user case studies and experiments.

1.2 Outline
The rest of this document is organized as follows: section 2 takes a look at related work, section 3 shows the approach taken when designing SEURAT_Edu, section 4 examines the architectural details of SEURAT_Edu, section 5 showcases SEURAT_Edu, section 6 discusses the conducted experiments, section 7 suggests avenues of future research, and section 8 provides a conclusion to the research.

2 Related Work
In order to address how successful this research is, the scope of what student learning entails must be refined. In this context, student learning has three main aspects: cognitive development, creativity, and critical thinking. Cognitive development is the most straightforward aspect since it relates directly to how students learn at different points in their careers. Meanwhile, creativity and critical thinking are more focused on the decision-making aspect of student learning. Finally, in order to make the best possible rationale documentation tool, existing software must be examined to determine what works best.

2.1 Cognitive Development
Cognitive development deals with the ongoing process of learning to think. The most applicable mapping of this process to what this project is trying to accomplish can be seen in Perry’s Scale (1999). This scale maps the developmental path that learners take, from basic duality where there is only right and wrong, to relativism where knowledge and correctness are contextually based on perspective. There are nine positions in total, but they can be broken up into three distinct categories: dualism, multiplicity, and relativism.

Dualism, stages 1 and 2, is characterized by the existence of all-knowing authorities, and right and wrong answers for everything. More explicitly, the authorities, which can be teachers, parents, or textbooks, always have the correct answer to any question, and any conflicting answers are wrong. At this stage, students think of intelligence as being able to regurgitate correct answers back to the authorities. It is only near the end of dualism that students begin to realize that a teacher’s role is also to help the students learn to think. One of the biggest signs of a student moving from dualism into the next stage, multiplicity, is the realization that maybe even the teachers have to search for their answers.

Multiplicity, stages 3 and 4, begins when students recognize that their teachers cannot provide a correct answer for every question. That being said, there may still be a belief that given enough time, the teacher will be able to find another teacher or textbook that can answer the question. When this doesn’t happen, though, and the teacher is unable to find any answer, the student moves onto the second phase of multiplicity where they believe any opinion is valid. In other words, the students believe that as long as their own opinions cannot be proven wrong, they are as just as right as any teacher’s opinion.

Relativism, stages 5 through 9, comes into play when students begin to consider the support provided alongside opinions. At this stage, the support provided with an opinion is
what determines how valid the opinion really is. This is an important progression because it marks a stage where students are able to evaluate information provided to them, even information from an authority, and decide how correct it is. This stage is a springboard through the last stages of Perry’s Scale where students learn to make their own decisions and commitments.

Most college freshmen enter at stage 2 or 3, which is when students begin to move from dualism to multiplicity (Wise 2004). It is up to the professor to provide an environment that supports the transition through Perry’s Scale. Robert Kloss (1994) talks about his experience working with college freshmen and cites experience when giving recommendations on helping them move from dualism into multiplicity. When it comes to helping his students grow, Kloss’s main idea is that “a nudge is better than a shove.” He believes that maintaining a high level of expectation, while providing a high degree of structure, is the best way to help students with the transition. More specifically, Kloss recommends subjecting the students to problems with ambiguous or multiple solutions. This forces the students to compare the solutions and provide support for or against each. A good mode for the discussion of solutions is peer to peer conversations. Kloss states that this works best because it “diminishes the instructor’s authoritative role” and puts more power into the students’ hands.

Bloom’s Taxonomy (1956) is another tool that can be utilized to help promote cognitive development. The first level of the taxonomy, knowledge, deals with the act of simply repeating information that was previously presented to the student. The next level, comprehension, is when students begin to compare information presented to them. Application is the stage where students use their understanding of the knowledge presented to them in order to make decisions. The fourth stage is analysis, which introduces the concept of providing evidence in support of claims. Synthesis is characterized by the process of combining separate ideas into a new structure, and providing alternative ideas. The final stage, evaluation, includes all the previous stages, and introduces a set of criteria that the students must make judgments against. Janet Burge and Bo Brinkman (2010) compared the ideas presented in Perry’s Scale and Bloom’s Taxonomy in terms of design rationale. Their work can be seen in Table 6.
Table 1 Bloom’s Taxonomy, Reflection and Rationale, and Perry’s Scale (Burge and Brinkman 2010).

<table>
<thead>
<tr>
<th>Bloom</th>
<th>Reflection and Rationale</th>
<th>Perry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge</td>
<td>Given a general decision problem, list and define the alternative solutions, as described in class. Example: What data structures can be used to store lists of items?</td>
<td>Students at all levels should be able to do this since it could be directly recalled from their lecture notes.</td>
</tr>
<tr>
<td>Comprehension</td>
<td>Given a general decision problem, and a set of criteria for making a selection, explain why the criteria are important. Example: Why is it important to be able to efficiently remove elements from a list of items? Given a set of alternatives for a general decision problem, differentiate between them. (This may require giving the students the criteria). Example: What is the difference between two data structures that store lists of items?</td>
<td>Students at all levels should be able to explain the criteria, but students still in the dualism stage may show biases toward certain alternatives (the “right” one) when differentiating between options and may not explore them in detail.</td>
</tr>
<tr>
<td>Application</td>
<td>Given a specific decision problem, give a list of possible solutions. Example: Given a design that requires sorting and searching a list of items, which data structures could be used to solve it?</td>
<td>Students in dualism may have difficulty providing more than one solution or more than one valid solution.</td>
</tr>
<tr>
<td>Analysis</td>
<td>Given a specific design problem, provide a list of possible solutions and map those to a set of design criteria. Example: Given a design problem that requires sorting and searching a list of items, list the appropriate data structures for storing the items and how they relate to criteria, such as time required to search, time required to add new items, etc.</td>
<td>Students in dualism may have difficulty providing more than one solution or more than one valid solution. If multiple solutions are produced, they may have trouble proposing arguments opposing solutions they have already deemed “correct” or identifying arguments supporting solutions other than the “correct” one.</td>
</tr>
<tr>
<td>Synthesis</td>
<td>Given a specific design problem, define the criteria that should be used in order to make a decision. Example: Given a design problem that requires sorting and searching a list of items, what criteria should be used to evaluate candidate data structures? Which criteria are more important to the specific problem?</td>
<td>This is not clear. Will students in dualism only come up with criteria that apply to their chosen alternative, discarding any criteria that do not support their beliefs? Will students in the multiplicity stage have issues identifying some criteria as being more important than others or will they consider all criteria equally valid?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Given a specific design problem, define alternatives and the key criteria, and use this information to select a solution. Example: A design problem that requires sorting and searching a list of items, what are the candidate data structures, what criteria apply in evaluating the appropriateness of each data structure to solving the problem, and given those criteria, which solution is the best choice?</td>
<td>Students in dualism are likely to have the same issues listed above, and are likely to have difficulty getting to the evaluation stage. Students in multiplicity may have difficulty in making a selection, even after identifying alternatives and criteria.</td>
</tr>
</tbody>
</table>
2.2 Creativity

One of the proposed benefits of rationale documentation is an increase in solution creativity. In order to consider this further, it may be helpful to limit the scope of the discussion. In terms of this project, a good place to start is with ideation, the process of generating and communicating new ideas. This is a critical part of the creative design process with which many students struggle. Burge and Brinkman (2010) claim that students feel “significant pressure to find the, or a, correct solution in as little time as possible.” This means that students may consider only one potential alternative before deciding to implement the solution, regardless of how appropriate the solution is.

Ray McCall (2013) also discusses ideation in his work, but he combines it with evaluation. In other words, McCall believes that “intertwining ideation and evaluation promotes creativity in software design because feedback about consequences of design decisions challenges designers to devise new ideas.” This concept clashes with Rittel’s (1966) classic ideas, where he saw no need to connect ideation and evaluation (McCall 2013). The idea of providing immediate feedback on student ideas could be one way to overcome students rushing into their first solutions. For example, if a student’s first thought was immediately contradicted by a professor’s insight, there would be a slim chance the student would follow through with that idea. Unfortunately there is the possibility that such strong influence from the professor would lead to fear of making suggestions. However, if this could instead be handled by providing feedback through an automated system, the students may be more comfortable to seek feedback.

Another aspect of creativity in software design is the idea of liminality. Carroll (2010) defines liminality in terms of creativity as “Thinking and acting on the border between two contrasting concepts or rules, such as rapid switching between convergent and divergent modes of thinking” (Burge and Brinkman 2010). They point out that this idea is “inherently opposite” from the concept of dualism, where students see only right and wrong (Perry 1999). Because most college freshmen fall within the dualistic stage, Burge and Brinkman suggest the students be forced to spend a predefined amount of time on ideation. They also suggest that “The time spent in the liminal phase of design, iterating from idea to evaluation and back, can be lengthened by the use of design rationale.”

2.3 Critical Thinking

Critical thinking is a term often associated with the field of computer science, but it is rarely explicitly defined. In terms of this project, Richard Paul gives a good start when he claims that “In thinking critically we use our command of the elements of thought to adjust our thinking to the logical demands of a type or mode of thought” (Paul 1990). Paul goes on to describe ten different elements of thought, which range from understanding the problem, to recognizing the differing viewpoints and alternatives, to understanding the implications of different solutions. It’s this idea of considering and analyzing multiple alternatives that this project, and computer science in general, are concerned with.
This definition of critical thinking lends itself well to the concept of argumentation. Andriessen defines argumentation as “a series, set or chain of arguments, linked by theme or reasoning,” (2003) where an argument is an expression used to support another expression. In this sense, an argument could be used to prove something, defend an idea, or “render more acceptable, believable, plausible.” One other important aspect of this definition is the fact that “reasoning” implies the generation of arguments, and not just the recitation of them. The link between argumentation and critical thinking is clear: both are used to evaluate, or provide, evidence for or against an idea. In terms of this project, all critical thinking is assumed to be argumentation-based.

2.4 Use of Rationale in Education (Argumentation-Based Rationale)

There have been many attempts to create software applications to aid in the collection of rationale. These applications have taken a variety of approaches to rationale collection, such as argumentation-based rationale, task-artifact framework, issue-based information systems and questions, options, criteria (Wang, et. al. 2010). This section will focus on argumentation-based rationale systems. Such systems are concerned with the “arguments about issues” in the design process (Shum and Hammond 1994).

The most obvious system to explore is SEURAT, which is the basis of this project (Burge and Brown 2008). The aim of SEURAT is to provide an integrated system for capturing argumentation-based rationale. This was accomplished by developing the application as a plugin for Eclipse, a widely used development environment. The system uses its own semi-formal representation of rationale, called RATSpeak, which is between informal conversation and machine readable text. There is a wide range of rationale types, including requirements, decisions, alternatives, arguments, and claims to name a few. What makes this system unique is that it inferences over the provided rationale and informs the user of instances where the most optimal alternative was not selected. This type of feature is great for students because it can help them understand the reasons why a specific alternative may be a better choice.

3 SEURAT_Edu Approach

SEURAT_Edu was developed in order to address the issue of documenting design rationale, based on Dr. Burge’s SERUAT. Both tools are aimed toward the goal of capturing design rationale, but there are several fundamental differences. The biggest difference is that SEURAT_Edu is a web application. Once an instructor adds the tool to their Niihka site, enrolled students can access it from any internet-connected computer. Compared to SEURAT’s tie to Eclipse, this makes the tool much easier for students in any field of study to use. The second difference is that SEURAT_Edu provides interfaces for instructors to create assignments for their students. These aspects make SEURAT_Edu ideal for a classroom setting. The rest of this section highlights the most important functions of the application.

3.1 Rationale Types

SEURAT_Edu uses the smallest subset of rationale types from SEURAT required to create meaningful rationale. This is because many of the rationale types from SEURAT are beyond the scope of what is typically required in a classroom assignment, such as tradeoffs
and questions, and providing support for every type would have significantly extended the required development time. The rationale items will be created in a tree format that allows the user to quickly navigate to or hide any pieces of rationale. The following rationale types are included in SEURAT_Edu:

- **Requirements** represent the software requirements of the system that the rational is being collected for, and can be functional or non-functional. Most often, arguments will refer to requirements in order to specify whether or not an alternative satisfies a requirement. Can also document the rationale for requirements.
  - e.g. Fast Random Access
- **Decisions** represent any kind of choice that must be made as part of the development process. Each decision can have either alternatives or sub-decisions, but not both.
  - e.g. Data Structure
- **Alternatives** represent possible solutions for a decision. Each alternative has arguments for and against it that are used to generate a score for the alternative. When creating alternatives, the user should not be filtering out solutions that they deem as bad, since that is the purpose of the tool.
  - e.g. Array, Linked List
- **Arguments** represent pros and cons of alternatives, or dependencies between alternatives. Each argument is about and contributes to its parent alternative while referring to a separate piece of rationale, like requirements, claims, or assumptions.
  - e.g. Alternative “Array” supports requirement “Fast Random Access”
- **Claims** represent criteria used in decision-making that were not explicitly given as requirements.
  - e.g. Easy to Code
- **Assumptions** represent things that the user believes to be true but is unsure about, or things that might change over time.
  - e.g. Already have User Profiles

### 3.2 Create Assignment

Because the application will be centered on classroom use, there needs to be functionality that allows the instructor to create assignments for the students. Additionally, the instructor needs to be able to provide starting rationale for each assignment so that the students have a starting point. By providing different amounts of starting rationale, the instructor can create assignments geared toward differing levels of Bloom’s Taxonomy. In order to create starting rationale, the instructor should first provide a fully completed solution rationale tree. Once satisfied, the instructor will copy the solution to a “student version.” From here the instructor can add, delete, or modify rationale in the student version. Any modifications will be visible to the students when they begin the assignment. Using this method, the instructor can create assignments at any level of Bloom’s Taxonomy by providing anywhere from no starting rationale to nearly complete rationale, as specified in Table 1.

The instructor should also be able to create group assignments where multiple students can concurrently work on a single rationale tree. The groups should be specified by the
instructor, and they should be able to use the same group for every assignment or create assignment-specific groups.

3.3 Complete Assignments
   Once the instructor creates an assignment, the students should be able to view the assignment details and complete it. When they first view the assignment, students should see the starting rationale provided by the instructor. Until the due date passes, students should be able to modify their rationale trees. Once the due date passes, students should still be able to view their rationale, but they should not be able to modify it. This lets the student go back through any previous assignments to review their work.

3.4 View Student Turnins
   At any point, the instructor should be able to view all of the current student turnins. This lets the instructor check to see how the assignment is going and also makes it easier for the instructor to help specific students. After the assignment due date has passed, the instructor needs to be able to see a complete list of all the student turnins for grading. All grading, though, cannot take place in SEURAT_Edu and must be recorded elsewhere by the instructor. At no point should the instructor ever be able to modify a student turnin.

3.5 Rationale Inferencing
   During the process of creating rationale, the tool should inference over the input and provide immediate feedback to the user. This feedback should warn the user of any issues with the rationale, like incomplete items, and should recommend alternatives based on the user input. If the recommended alternative is not selected, the user should be notified in order to make them think about why they selected the alternative that they did. These status items must be very apparent and explicit in what the stated issue is.

4 SEURAT_Edu Architecture
   The driving idea behind SEURAT_Edu was to make an easily accessible application that students from any field of study can use. Instead of coupling the application closely to an existing software, a web-based solution was chosen. By creating an application that lives on a server and is accessible via a browser, users do not have to be familiar with any external software. The only downside is that this requires a publicly accessible web server to deploy the application. This section outlines how SEURAT_Edu is implemented at Miami University, and explains how it could be utilized by other universities.

4.1 Overview
   An overview of the SEURAT_Edu architecture is shown below in Figure 1.
The SEURAT_Edu infrastructure is as follows:

- **Apache Server**: Public web server where the SEURAT_Edu application lives.
- **SEURAT_Edu**: Web application for documenting design rationale.
- **SEURAT_Edu Database**: Relational database that holds all assignment and rationale data for SEURAT_Edu.
- **Niihka**: Miami University’s learning management system that interacts with SEURAT_Edu.
- **Course Site**: An individual course’s page within Niihka, which SEURAT_Edu has been added to.
- **Tool iFrame**: SEURAT_Edu is displayed within an iFrame on the course site. What this means is that the application is displayed within Niihka, while being hosted remotely on the apache server.

### 4.2 Apache Server

In order to access SEURAT_Edu within a browser, the application must be hosted on a publicly accessible web server. The server used throughout the development and testing phases of SEURAT_Edu was provided by Dr. Campbell at Miami University via an OpenStack cloud. OpenStack, which is an open-source cloud computing platform, was used to create an Ubuntu Linux server specifically for the SEURAT_Edu project. The server must have Apache Tomcat and MySQL installed in order to run the application, which can be done easily at the command line. Once these packages have been installed, the user can deploy the SEURAT_Edu application by copying the .war file, created as described below, into the webapps directory.

At this point SEURAT_Edu is deployed to the server and is ready to be accessed through Niihka. However, without an HTTPS connection using a certificate from a Certificate Authority (CA), Niihka denied any requests to and from SEURAT_Edu’s server. This was addressed with additional help from Dr. Campbell in receiving a certificate. After installing the certificate and enabling HTTPS, the server was ready to communicate with Niihka.
4.3 SEURAT_Edu

SEURAT_Edu was developed using Google Web Toolkit (GWT). According to the official site, “GWT is a development toolkit for building and optimizing complex browser-based applications” (GWT 2014). It does this by letting developers write Java code that compiles into efficient, standalone JavaScript code. GWT provides numerous widgets and APIs for constructing web-pages, and automatically handles browser compatibility issues for the developer. In order to handle asynchronous applications that update without reloading the page, GWT utilizes remote procedure calls (RPCs) to communicate with a server backend. In short, it allows a Java developer to very easily create clean, cross-browser web applications without any prior web development experience. After compiling the GWT project, the developer is left with a .war file which can be easily deployed to a web server.

For all of these reasons, GWT was an obvious choice for SEURAT_Edu. Because the timeline for the entire project was rather short, not much time could be dedicated to learning new languages and web development practices. Also, because knowledge of Java is more ubiquitous than knowledge of web development, at least at Miami, there is a greater chance of the project being picked up and continued by students in the future.

4.4 SEURAT_Edu Database

In order to store assignment and rationale data, SEURAT_Edu utilizes a MySQL database. The database lives on the same server as the application and is accessed via RPCs in the GWT code.

4.4.1 Tables
A complete list of the MySQL database tables used by SEURAT_Edu can be found in Appendix A.

4.5 Niihka Learning Management System (LMS)

Miami uses its own version of the Sakai LMS (Sakai 2014), called Niihka, to handle online course management. When accessing external tools, Niihka sends a defined list of parameters to the tool full of course and user identifying information. All of this data is part of the Learning Tools Interoperability (LTI) Standards (Learning Tools Interoperability 2014).

4.5.1 LTI Standards
The LTI Standards, managed by IMS Global Learning Consortium, is a specification for communication between an LMS and a third party external tool. What this means behind the scenes is that when an LMS contacts an external tool, the LMS sends a defined list of parameters to the tool which contains course and user identifying information. What this enables the third party organizations to do is make a single tool that can interact with any LMS that supports the LTI Standards. Because an end goal of SEURAT_Edu is to expand to other universities, the integration with these standards is a crucial aspect of the tool. Even as Dr. Burge moves to new universities and continues her research, she will be able to use SEURAT_Edu at any of the universities that support the LTI Standards.
There is a list of about 20 separate parameters passed to the tool each time it is accessed, but SEURAT_Edu only uses five of them:

- **user_id**: The unique identifier of the user logged into the LMS. For Niihka, this is the 8 character username that students and instructors use to login to Miami sites.
- **lis_person_name_full**: The user’s first and last name. This value is used primarily by the instructor to identify student turnins.
- **roles**: Denotes whether this is a student or instructor, and determines which layout to display. Within Niihka, instructors are denoted by “Instructor” and students are denoted by “Learner.”
- **context_title**: The display name of the course from which the tool is being accessed.
- **context_id**: The unique identifier (GUID) of the course from which the tool is being accessed. This is needed in conjunction with the display name since the same course is often offered in both the spring and the fall.

### 4.5.2 Problems Encountered

Though the LTI Standards make SEURAT_Edu vastly more accessible, there were several issues that needed addressed. The first was related to the use of GWT and how the parameter list is passed to SEURAT_Edu. When Niihka passes the parameters, it does so via a POST request which must be handled by the server side of an application. Because GWT creates client side applications, it is not possible to directly access the POST parameters from within SEURAT_Edu. In order to address this issue, a “Redirect Servlet” was created. Instead of pointing Niihka directly to SEURAT_Edu, it instead points to a Java servlet which reads the parameters and stores the values in the session. The servlet then redirects to the SEURAT_Edu application, which reads and clears the values from the sessions. Though this may be common web development practice, this was one of the biggest hurdles in development of SEURAT_Edu due to inexperience with web development.

The second limitation of the LTI Standards has to do with the LMS course rosters. There is no way to automatically send the course rosters to external tools, which affects the group functionality within SEURAT_Edu. Within Niihka there are several options to send additional pieces of information, but none of them pass in the roster. Several ideas were considered to address this but none of them were adopted. The first was to have the instructor add the roster in as a custom parameter, and the second was to populate the roster as students first access SEURAT_Edu from Niihka. Neither of these seemed to effectively solve the problem, which is why a roster creation functionality had to be added to SEURAT_Edu.

### 5 SEURAT_Edu Implementation

The currently implemented SEURAT_Edu has the following features:

- Can be added to a Niihka course site.
- Home screen presents all course assignments, and provides easy access to assignment information, turnins, etc.
• Instructor can provide a course roster, used for group assignments.
• Custom student groups for each assignment.
• Course assignments that allow instructor to provide starting rationale.
• Rationale tree for representing rationale relationships and dependencies.
• Editor for creating and modifying rationale.
• Status panel that provides immediate feedback on user input.

5.1 Adding SEURAT_Edu to Niihka Site

Before SEURAT_Edu can be used, the course instructor must add the tool to their Niihka site. This is a very easy process that can be completed in just a few minutes by following Niihka’s instructions for adding an external tool. All the instructor needs to know is the URL where SEURAT_Edu can be accessed.

5.2 Home Screen

Figure 2 shows how the home screen looks when an assignment has been created.

![Instructor home screen](image)

Figure 2 Instructor home screen

The home screen is the first screen the user sees when accessing SEURAT_Edu. This is where students and instructors can view assignments and navigate the tool. At the very top of the page, the user will see a breadcrumb telling them from which class SEURAT_Edu is being accessed. Right below the breadcrumb are several self-descriptive administrative buttons. The “Create Assignment” button takes the instructor through the process of creating an assignment. The “Groups” button allows the instructor to create groups for the current course. And finally, the “Roster” button takes the instructor to a screen where they can upload a class roster.

In the center of the screen is where created assignments are displayed. On the first load of the tool, this section will be blank since no assignments have been created. Figure 1a shows how each assignment is displayed. In this example, “VISE” is the assignment name. Underneath the assignment name, there are several options that the instructor can choose from as well as the due date of the assignment. The “Edit” option takes the instructor to the assignment creation page where they can modify the assignment.
Selecting “Delete” will display a popup asking the instructor if they would really like to delete the assignment, as seen in Figure 2. If they select yes, the assignment will be deleted from the course and will no longer be available for viewing or editing.

The “Turnins” button takes the instructor to a list of the current student turnins as seen in Figure 3. Each student is automatically added to this list the first time they access the assignment. This means that if a student’s name does not appear here, they have never accessed the assignment. From here, the instructor can click on any student’s name to be taken to that student’s current rationale tree. When viewing student turnins, the instructor is only able to look at the student data and is unable to modify any of the students’ rationale.

Finally is the “Export All” button. At this point in time, this feature has not yet been completed. What this will ultimately let instructors do is convert each student’s rationale into a text format and store the file locally on the instructor’s computer. This would provide instructors with a way to save rationale indefinitely and view the turnins offline.

5.3 Roster
Selecting the “Roster” button on the home screen brings the instructor to the roster creation page. One limitation of Niihka is that there is no way to pull the class roster data into SEURAT_Edu. The only time that the roster data is used is when creating class groups. Therefore, if the instructor does not intend to do group assignments, they can ignore this aspect of the tool. When first accessing this page, the instructor will see a blank textbox on the left, and several buttons.
As seen in Figure 4a, the textbox is where the instructor inputs the class roster. The names used for the roster must be the unique IDs of the students. This textbox is flexible in how it accepts data. Names can be separated by commas, semicolons, or any kind of whitespace, and email addresses will be split on the ‘@’ symbol and will take the left hand side. For example, the “john5@school.edu” in Figure 4a will be stored as “john5” in the database. Most LMSs, including Niihka, provide a way for instructors to download the class roster. Instructors should be able to simply open the downloaded file, copy the roster from excel or whatever text editor is opened, and paste into this textbox. Upon clicking the “Add” button, the textbox will be cleared and the roster will be displayed on the right hand side as seen in Figure 6 below.
At this point, all the names on the right side have been stored in the database for this class. If there was an error, or if a student drops the class, they can also be removed from the roster here. Simply select their checkbox and click the “Remove Selected” button to remove them. The “Select All” button can be used in conjunction with the “Remove Selected” button to completely clear the class roster. The instructor can add or remove students from their class any number of times.

5.4 Groups
Figure 7 shows the interface for creating groups within SEURAT_Edu.

The “Groups” page is where the instructor uses the roster data to create groups for the class. Because each assignment may utilize different groups, instructors can multiple groups of groups, or clusters of groups. As seen in Figure 5a, the roster that was previously created is now displayed here. At the top of the group area, there is a textbox and a dropdown. The textbox is where the instructor specifies a unique name for this cluster of groups. The accompanying dropdown is where the instructor can see all previously created clusters. Selecting one from the dropdown will load that cluster of groups onto the page for editing.
Underneath the roster, there is a “Group” dropdown that is initially populated with just “New Group”. Once the instructor selects the students from the roster that will be in the same group, the instructor clicks the “Add” button to create the group. As seen in Figure 5b, the students are moved into their groups and removed from the roster on the left. Also, the “Group” dropdown has been populated with numbers to denote the new groups. Now the instructor can either add students to “Group 1” by selecting “1” from the dropdown, or the instructor can create a new group by selecting “New Group” from the dropdown. These clusters do not automatically save each time a group is modified, so the instructor must select the “Save” button once finished. The “Delete” button deletes the current cluster of groups.

To remove students from a group, select their checkboxes and click the “Remove Members” button. This will remove the selected students from their groups and move them back into the roster section on the left. Any groups that lose all their members are automatically removed from the list of groups and the group numbering is adjusted accordingly. Instructors can create any number of groups and not all students are required to be included in a group.

5.5 Assignment Creation

When creating an assignment, the instructor has many things to consider. Not only does the instructor have to specify the usual items such as title, description, and due date, but they also have to decide how much starting rationale to provide for students.
Figure 9 Instructor assignment creation page

Figure 9 shows everything that an instructor must consider when creating an assignment. The assignment title is what will display on the home page to identify this assignment. The description field can be used to provide instruction to students, but is not required. The current implementation of the due date simply requires the instructor to select a date and the assignment will be due on midnight of the specified day. After the due date is a checkbox where the instructor can make the assignment available. Only when this box is checked can students see and have access to this assignment. This option lets the instructor work on any number of assignment at once without worry that the students will see them before they should be able to. This box can only be checked once the instructor has created a student version of the assignment, as described below. Finally, the instructor can specify whether or not this is a group assignment. Selecting the “Individual” tab of the panel makes this an individual assignment. Selecting the “Group” tab, however, lets the instructor choose a cluster of groups to use for this assignment. Group work is still under development and currently does not support concurrent use. Though multiple group members can view and work on rationale at the same time, they will not receive live updates to reflect their group members’ work, and it is possible that concurrent work will ruin their rationale. This is one of the biggest focuses for future work.

Once all these fields have been completed, the instructor is ready to create rationale for this assignment. Selecting “To Solution” will save the assignment and take the instructor to the rationale editor where they can create their solution. Any rationale created here is not
directly visible to students, so the instructor should create as complete a rationale tree as possible. Once satisfied with the solution, the instructor can revisit the assignment details page from the home screen. Now the instructor is ready to create the starting rationale template for the students.

![Figure 10 "Create Student Version" warning prompt](image)

Selecting the “Create Student Version” button will display the prompt in Figure 10. As stated, this creates an exact copy of the solution rationale and can only be done once. Selecting “Continue” takes the instructor back to the rationale editor where they should see the exact same rationale as their solution. From here, the instructor can modify the rationale tree as much or as little as they like, and it will not affect their solution. The most common practice is to delete the pieces of rationale that the students should be responsible for completing. Once satisfied with the student version, the instructor can navigate back to the assignment details, check the “Make available to students” box, and save the assignment. At this point, the assignment is ready to be completed by the students. When they begin the assignment, they will receive an exact copy of the instructor’s student version.

### 5.6 Rationale Tree

In order to display the user’s rationale, all items are added to the rationale tree. As users modify their rationale, the tree is populated to reflect the changes. Each type of rationale is represented in the tree by a colored icon, as seen in Figure 11.

![Figure 11 Rationale Tree](image)

These icons are designed to provide the user with a quick way to see what is happening within the rationale. For example, requirements can be classified as functional, dark blue, or nonfunctional, light blue. As seen on the requirement and decision, each icon is also updated
to reflect any status items for that piece of rationale. One of the most important aspects of alternatives is whether or not they have been adopted, which is denoted by the sigma on the icon. For arguments, the most important thing to be able to see at a glance is whether or not it is for against the alternative. In Figure 11, the argument is outlined in red which means it is against the alternative. Arguments outlined in green are for the alternative. Finally, assumptions and requirements can be disabled, which is denoted by a small “D” in the upper right corner. This can be seen on the assumption above.

Items in this tree can be expanded or collapsed in order to show only specific parts of the tree. Because the information is organized into a tree, the user utilizes parent/child relationships in order to model their rationale. According to what type of rationale is currently selected, buttons are displayed that allow the user to create child items for the selected rationale. Each of the types of rationale have very specific functions and input fields.

5.7 Rationale Editor

Figure 12 shows the main interface for editing rationale.

Once the student or instructor enters the rationale editor, they will be presented with the screen shown in Figure 12. This screen is comprised of 4 distinct panels, each with a distinct use. The north panel contains buttons for navigating SEURAT_Edu, creating rationale, and toggling the help display. The west panel is where the rationale tree lives, which is how the user navigates through their rationale. As labeled, the south panel contains warnings and errors related to the rationale tree. Finally, the center of the screen is where the user will input provide input for their rationale. Each of these screens are scrollable and resizable so that the user can show or hide any section they want.
5.7.1 Requirements

Figure 13 represents the fields involved in creating a requirement.

In order to create a requirement, select “Requirements” in the rationale tree and click the “Create Requirement” button. The fields are as follows:

- **Name**: This field denotes how the requirement is displayed and referred to within the rationale tree. There are no constraints on this value, except that it must be present.
- **Description**: Use this space to provide an optional explanation of the requirement. There are no constraints on this value.
- **Type**: Requirements can be one of two types within SEURAT_Edu: functional, or non-functional. This value does not affect how the requirement interacts with other pieces of rationale. Cosmetically, functional requirements are represented by a dark blue icon, while non-functional requirements are represented by a light blue icon.
- **Importance**: This dropdown specifies how important the requirement is, ranging from “Not” to the default value of “Essential.” The value of this dropdown can be used by arguments, and is very important to the inferencing process.
- **Status**: The status field is used to denote the current status of the requirement. The values are: Undecided, Addressed, Satisfied, Violated, Retracted, and Rejected. The user can manually select a value, or leave it at the default “Undecided” value. If “Undecided” is selected, SEURAT_Edu will automatically update the dropdown to reflect what the arguments are saying, such as “Satisfied” or “Violated”.
- **Enabled**: Because requirements are always changing, they can be toggled on or off within SEURAT_Edu. Leaving this box checked will include any arguments that reference this requirement in the inferencing. Deselecting this box will cause
inferencing to completely omit any arguments referring to this requirement. When deselected, the requirement’s icon in the rationale tree will update to include a small “D” to represent that it is disabled.

In the north panel, the user can select to create an argument for the currently selected requirement. Creating arguments for a requirement is not a feature that most users will utilize. Arguments created in this way are strictly for the user’s efficacy and are not used in the inferencing process. As arguments are created for and against requirements, the corresponding fields within the requirement editor are updated accordingly.

5.7.2 Decisions

Figure 14 represents the fields involved in creating a decision.

![Figure 14 Creating a decision](image)

In order to create a decision, select “Decisions” in the rationale tree and click the “Create Decision” button. The fields are as follows:

- **Name**: This field denotes how the decision is displayed and referred to within the rationale tree. There are no constraints on this value, except that it must be present.
- **Designer**: Optionally specify who came up with this decision. This is typically only used when working on group assignments.
- **Description**: Use this space to provide an optional explanation of the decision. There are no constraints on this value.
- **Type**: There are two types of decisions within SEURAT_Edu: single choice and multiple choice. By selecting single choice, the user is specifying that only one alternative should be selected and that if more than one is selected, then an error
should be reported. Choosing multiple choice means that any number of adopted alternatives is acceptable.

- **Status**: The status field is used to denote the current status of this decision. The values are: Addressed, Resolved, Unresolved, Non-Resolvable, Retracted. This value defaults to “Unresolved” and can be manually set by the user.

- **Sub-Decision Required**: Use this checkbox to denote whether this decision should have child decisions or child alternatives. By checking the box, the user is specifying that this is a multi-tiered decision and that it should explicitly have child decisions. Leaving the box unchecked specifies that this decision should only have child alternatives. If the user tries to switch this value while it has children, they will be notified that they must first delete the children before proceeding.

Depending on whether or not the “Sub-Decision Required” box is checked or not, the user will be able to create child alternatives or decisions from the north panel. As the user provides alternatives for this decision, the “Alternatives” box will be populated with the names of the alternatives and each of their scores. When decisions are initially created, the status panel will display an error because no alternative has been selected, as seen in Figure 15 below.

### 5.7.3 Alternatives

Figure 15 represents the fields involved in creating an alternative.

![Figure 15 Creating an alternative](image)

In order to create an alternative, select an appropriate decision and click the “Create Alternative” button in the north panel. The fields are as follows:
- Name: This field denotes how the alternative is displayed and referred to within the rationale tree. There are no constraints on this value, except that it must be present.
- Designer: Optionally specify who came up with this alternative. This is typically only used when working on group assignments.
- Description: Use this space to provide an optional explanation of the alternative. There are no constraints on this value.
- Status: This dropdown is used to specify whether or not the alternative has been selected. The values are: Adopted, At Issue, and Rejected. Choose “Adopted” to specify that this alternative has been selected. Both of the other options mean that the alternative has not been selected, but for different reasons. Use the default value of “At Issue” to specify that an alternative needs more attention, and use “Rejected” to simply specify that it has not been chosen.
- Score: This value is updated automatically as the user creates arguments about this alternative, and should be strongly considered when selecting an alternative. According to the input provided by the user, the most positive scoring alternative is the best and the most negative scoring alternative is the worst.

In the north panel, the user can select to create an argument about the currently selected alternative. Every alternative should have arguments about it. As the user creates arguments, their names will display in the corresponding for or against field.

5.7.4 Arguments
Figure 16 represents the fields involved in creating an argument.
To create an argument, select a requirement or alternative and click the “Create Argument” button. The fields are as follows:

- **Name**: This field denotes how the argument is displayed and referred to within the rationale tree. There are no constraints on this value, except that it must be present.
- **Designer**: Optionally specify who came up with this argument. This is typically only used when working on group assignments.
- **Description**: Use this space to provide an optional explanation of the argument. There are no constraints on this value.
- **Argument Type**: Use this dropdown to select what type of rationale this argument is referring to. The values include each type of rationale, except for decisions. As this value changes, the “Argues” dropdown is automatically populated with rationale items of the corresponding type. The “Type” dropdown is also populated with the appropriate options as this value changes.
- **Argues**: Use this dropdown to select which piece of rationale this argument is referring to. As the “Argument Type” changes, this dropdown is automatically populated with all items of the corresponding type. The only exception is if the argument type is specified as alternative: because an argument cannot make a circular reference to its own parent alternative, the argument’s parent will not be present as an option.
- **Type**: This is how the user specifies whether this is a positive or negative argument. Depending on the argument’s sentiment, the argument’s icon within the rationale tree updates with green for positive or red for negative. The dropdown is populated with different options depending on the “Argument Type” selected:
  - For requirements, the options are: Satisfies, Violates, and Addresses. Both “Satisfies” and “Addresses” have a positive sentiment, while “Violates” is negative. If the referenced requirement’s status is set to “Undecided,” it will update to reflect the value of this dropdown.
  - For alternatives, the options are: Pre-Supposes and Opposes. Arguments referring to alternatives act differently than other arguments in several ways. Most notably, they do not contribute to the score of and do not count as being for or against their parent alternatives. Instead, these arguments are used to represent dependencies between alternatives. For example, assume that in order for alternative X to be selected, alternative Y must also be selected. In this case, the user would create an argument about X that references, pre-supposes, Y. However, assume that in order for X to be selected, Y must not be selected. In this case, the user would create an argument about X that references, opposes, Y.
  - For claims and assumptions, the options are: Supports and Denies. “Support has a positive sentiment and “Denies” has a negative sentiment.
- **Importance**: This dropdown specifies how important the argument is, ranging from “Not” to “Essential.” However, the default value, “Default,” pulls the importance value from the referenced rationale instead. What this lets the user do is modify the
referenced item’s importance value and see how it impacts the inferencing across all the arguments.

- **Amount**: This value is used to arbitrarily compare alternatives on a scale from 1-10, which makes it one of the more difficult fields to become familiar with. Assume there are two alternatives, “Array” and “Linked List,” and both have an argument referencing the requirement “Fast Random Access.” Though random access is possible with both alternatives, the array is obviously much faster. For this reason, the “Array” alternative might set its amount to 10, and the “Linked List” alternative might set its amount to 1. The amount is the most important factor in the magnitude of each argument’s score and should be given significant consideration.

- **Plausibility**: This dropdown is only for the user’s efficacy and has no impact on inferencing. The options are: Low, Medium, High, and Certain.

Arguments are different than requirements, decisions, and alternatives in that they cannot have children. Though users can create claims and assumptions from the north panel, they will go into their own section and will not be created as children of the argument. That being said, each argument’s referenced rationale is displayed under the argument within the rationale tree for ease of access.

### 5.7.5 Claims

Figure 17 represents the fields involved in creating a claim.

![Figure 17 Creating a claim](image)

To create a claim, select an argument or “Claims & Assumptions” and click the “Create Claim” button. The fields are as follows:

- **Name**: This field denotes how the claim is displayed and referred to within the rationale tree. There are no constraints on this value, except that it must be present.

- **Description**: Use this space to provide an optional explanation of the claim. There are no constraints on this value.

- **Importance**: This dropdown specifies how important the claim is, ranging from “Not” to the default value of “Essential,” defaulting at “Moderate.” The value of
this dropdown can be used by arguments, and is very important to the inferencing process.

Claims are pretty straightforward, but have a twist. Unlike other rationale types, claims should follow a naming convention due to how arguments refer to them. For example, assume the user has alternative “Linked List.” However, the user is not familiar with linked lists, so they make a claim named “Hard to do.” When creating the argument that refers to “Hard to do,” the user runs into an issue: they want to support claim “Hard to do” but this gives the argument a positive sentiment, and score, when it should have a negative score since this is a bad thing. Instead, the user should name the claim “Easy to do” and then deny the claim in order to correctly create a negative sentiment, and score.

5.7.6 Assumptions

Figure 18 represents the fields involved in creating an assumption.

To create an assumption, select an argument or “Claims & Assumptions” and click the “Create Assumption” button. The fields are as follows:

- **Name:** This field denotes how the claim is displayed and referred to within the rationale tree. There are no constraints on this value, except that it must be present.
- **Description:** Use this space to provide an optional explanation of the claim. There are no constraints on this value.
- **Enabled:** Because assumptions are always changing, they can be toggled on or off within SEURAT_Edu. Leaving this box checked will include any arguments that reference this assumption in the inferencing. Deselecting this box will cause inferencing to completely omit any arguments referring to this assumption. When deselected, the assumption’s icon in the rationale tree will update to include a small “D” to represent that it is disabled.

Assumptions should follow the same naming convention as claims, for the same reason. One of the more interesting uses of assumptions is to represent several possible values for a detail of the project. For example, consider a user is creating a piece of software but is
unsure of how many people will use it. The user could create several assumptions, each of which represents a certain number of customers. Then within an alternative the user could create a separate argument, with different values, for each assumption. After this, the user could enable one assumption at a time to see how inferencing changes for different numbers of customers.

5.7.7 Help Panel

Figure 19 shows what the rationale editor looks like when the help panel is opened up.

![Figure 19 Rationale editor help panel](image)

The help panel can be toggled open to display information about the fields when creating rationale. As the user selects different types of rationale, the help panel updates to provide the appropriate help text. The panel is scrollable and manually resizeable.

5.8 Status Panel

Figure 20 gives an example of an error and a warning as displayed in the status panel.

![Figure 20 Status Panel](image)

Every time users create, modify, or delete rationale, SEURAT_Edu inferences over the input to provide immediate feedback to the user. All status items are listed in the status panel as shown in Figure 20. Each status item has an icon to distinguish the type, either warning or error, and a short description. Each description gives a short explanation of what the problem is, and has a bolded rationale name to specify which piece of rationale has the status. Clicking on a status item opens the rationale tree to the bolded item and opens the rationale
editor for that item so that the user can easily address errors. Also, the rationale tree icons of any items with statuses are updated to reflect any warnings and errors.

5.8.1 Inference Types

Table 2 provides a list of all the inferencing done within SEURAT_Edu. Each status type is classified by the message displayed within the status panel, the type of rationale that the status applies to, and the error level. Within the error text of each status item, the bolded text represents the rationale that the status belongs to.

Table 2 Rationale inference types

<table>
<thead>
<tr>
<th>Error Text</th>
<th>Rationale Type</th>
<th>Error Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected alt &lt;name&gt; violates requirement &lt;name&gt;</td>
<td>Requirement</td>
<td>Error</td>
</tr>
<tr>
<td>No alt selected for decision &lt;name&gt;</td>
<td>Decision</td>
<td>Error</td>
</tr>
<tr>
<td>Decision &lt;name&gt; has too many selected alts: &lt;name&gt;, &lt;name&gt;</td>
<td>Decision</td>
<td>Error</td>
</tr>
<tr>
<td>Alt &lt;name&gt; selected but &lt;name&gt; is better for decision &lt;name&gt;</td>
<td>Decision</td>
<td>Warning</td>
</tr>
<tr>
<td>Alt &lt;name&gt; selected for decision &lt;name&gt; has no args in its favor</td>
<td>Decision</td>
<td>Error</td>
</tr>
<tr>
<td>Alt &lt;name&gt; of &lt;name&gt; requires non-selected alt &lt;name&gt;</td>
<td>Decision</td>
<td>Error</td>
</tr>
<tr>
<td>Alt &lt;name&gt; of &lt;name&gt; is opposed by selected alt &lt;name&gt;</td>
<td>Decision</td>
<td>Error</td>
</tr>
<tr>
<td>Duplicate args found regarding alt &lt;name&gt;: &lt;name&gt;, &lt;name&gt;</td>
<td>Alternative</td>
<td>Warning</td>
</tr>
<tr>
<td>Contradictory args found regarding alt &lt;name&gt;: &lt;name&gt;, &lt;name&gt;</td>
<td>Alternative</td>
<td>Error</td>
</tr>
</tbody>
</table>

6 Experiments

As SEURAT_Edu was being developed, it was used in several Miami University courses to accompany actual class assignments. Each of the assignments directly related to the course material and lent themselves well to the use of SEURAT_Edu. Leading up to the assignments, the instructors had one on one training with SEURAT_Edu before attempting to create their assignments. In each assignment, the instructor provided starting rationale that the students were expected to complete either on their own or in groups. On the day of the assignment, an in-class demonstration of SEURAT_Edu was given to the students to help
them learn how to use the tool. After the assignments were completed, the student data was analyzed in an attempt to determine the effectiveness of SEURAT_Edu.

6.1 Design

Each assignment had one in-class day that consisted of four distinct parts:

- **Introduction/Waiver:** At the beginning of the class period, the instructor would introduce the assignment and give a brief explanation of why SEURAT_Edu was being used. Afterward, the students received waivers that they could optionally sign to allow their data to be analyzed. It was made very clear to the students that whether or not they released their data would have no impact on their grade for completing the assignment. Also attached to the waiver was a demographic survey.

- **Demonstration:** In addition to the actual class assignment, a demonstration assignment was created to teach the students how to use SEURAT_Edu. The demonstrations consisted of walking the students through accessing the tool, viewing the assignment, and modifying rationale. The demonstrations were conducted using a projector, while the students watched, asked questions, and made suggestions for the demonstration assignment.

- **Experiment:** After the demonstration, the students were let loose on the assignment. As they worked, the instructor walked around to help answer questions about the assignment and to address any questions about SEURAT_Edu. It was not required that students finish the assignment during the class period.

- **Survey:** Students received usability surveys when there were only 10-15 minutes left in the class period. These surveys were used to gather student feedback on what aspects of the tool worked well and what aspects still needed work. All of this data was recorded regardless of whether or not the waiver was signed.

6.2 CSE274 – Data Structures

In fall of 2013, SEURAT_Edu was used inside a data structures course at Miami University. This is a 200 level course consisting almost entirely of second year CS students, with a handful of third and fourth year CS students, as shown in Figure 21. There was only one student that was not taking this course to satisfy a requirement. The project used in conjunction with SEURAT_Edu was to code a game, VISE, as specified by the instructor. This assignment provided a unique challenge to students because the game board was composed of hexagonal spaces, which led to the tough decision of how to represent the game board in their code.
The instructor identified several data structures that he felt the students may attempt to utilize: arrays, adjacency lists, and graphs (nodes and edges). Using this information, the instructor generated a complete rationale tree that represented all the decisions the student would have to make, as well as some alternatives and arguments that the instructor felt were applicable. After completing his own rationale tree, as seen in the left hand side of Figure 22, the instructor created a version that the students would have to complete. The student version was an exact copy of the instructor’s tree, except no alternatives were selected and most of the argument fields had been cleared, as seen in the right hand side of Figure 22. This meant that students would have to fill out the argument fields and then choose alternatives based on their arguments.

The assignment was designed in this way for several reasons, mostly related to where in the curriculum the course sits. Because the course is a prerequisite for most other courses in the computer science curriculum, it can be expected to consist of mostly second year students going through the multiplicity stage of Perry’s Scale. At this stage, students are struggling with the idea that teachers do not always have a correct or best answer. For this reason, the
instructor made it very clear that each of the alternatives he provided was viable and that there were many alternatives he had not even considered. The next stage in Perry’s Scale is relativism, which is when students begin to focus on the support provided for and against opinions instead of who provides the opinions. This helped drive the idea behind providing students with incomplete arguments: though students may struggle with identifying valid arguments, they can still get the experience of deciding how valid and important specific arguments are. By giving students this experience, the hope is that they will be more comfortable constructing arguments in the future.

6.2.1 Demonstration

CSE274 is a data structures class, so it only made sense to provide a demonstration centered on data structures. As a class, the students were asked to decide upon a data structure to represent a calendar. The starting rationale that they were provided with was very minimal; it consisted of two requirements specifying the need for random and sequential access of dates. The students provided suggestions, which were then entered on the projector. This demonstration was interactive and was completely reliant on the students’ input.

6.2.2 Analysis

The results of the assignment can be found in Figure 23, which represents the number of students from each year of study that chose each type of data structure. It is evident that the younger students had more variation in their adopted solutions, but the reason behind this is hard to decipher.

![Adopted Solutions by Year of Study](image)

Figure 23 Adopted Solutions

The most obvious conclusion is that this is simply a result of not enough data points. Because there are more second year students than fourth year students, there is bound to be greater variation. The natural question is whether or not there would be similar variation in fourth years given more fourth year students, which is unfortunately impossible to determine from this data. Another possibility, though, is that these younger students had greater variation simply because they had trouble identifying the best alternatives. Because the students were given control of how important each argument was, there were bound to be
differences. For example, one student may have found familiarity to be very important, while another student was more focused on efficiency.

Digging into the students’ rationale provides an interesting insight. When the students were completing the assignment, they were supposed to adjust the argument amounts to whatever they thought was appropriate. However, nothing was stopping the students from leaving them at the default values. Figure 24 represents on average for each adopted solution how many argument amount fields were changed from the default value. For example, students that opted to use an array typically changed the amount field of about seven different arguments. On the other hand, students that chose adjacency lists or graphs averaged only .5 changes. What this means is that the students that opted for adjacency lists or graphs had probably already made up their minds and simply filled out rationale in order to complete the assignment.

![](image)

**Figure 24 CSE274 Arguments Changed**

One of the most important aspects of SEURAT_Edu is the ability to provide real-time feedback for the students in the form of status messages. Though these status messages were originally intended solely for the students’ efficacy, they have proved incredibly useful in helping instructors gauge how comfortable students feel with the material. As shown in Figure 25, for example, there were several instances of a student selecting an alternative that was, according to the provided arguments, inferior to another alternative. Specifically, three students opted to use either an adjacency list or a graph when their arguments pointed toward an array as the optimal choice. The only explanations for this are that the student had already made up their mind or that they were simply not comfortable using an array-based solution.
This assignment was a good introduction to SEURAT_Edu for the students, but it did not produce much meaningful data. Because of the way the assignment was structured with nearly-complete rationale tree, the students did not have to add very much of their own rationale. Only 8 new arguments, 5 of which were about an alternative being easier to code, were created across the 16 students that completed the assignment. Similarly, 6 of the 16 students did not even update the argument amount fields when completing the rationale. All of this is most likely due to the structure of the assignment. When conducting experiments in the future, this type of assignment should be avoided if looking for meaningful research data.

6.2.3 Survey

Because this was the first time SEURAT_Edu had ever been used in a class, most of the survey questions were focused on the usability of the tool. The actual survey document used in this experiment can be found in Appendix D. As seen, all the questions were related to how easy it was to carry out some action within SEURAT_Edu, like viewing an assignment or creating rationale. These data points were simply used to refine the tool, and therefore do not provide any meaningful statistics related to the original research questions.

There were, however, two open-ended questions at the end of the survey asking the students for general positive and negative feedback about SEURAT_Edu. Most responses were still geared toward the usability of the tool, but several students used this opportunity to speak on the concept of documenting their design rationale. According to one student, “Writing out all the design decisions beforehand helps you think things through.” Another student stated, “…you can see how such design decisions impact how you must design the rest of the project.” Responses like these are very encouraging considering the fact that documenting design rationale is additional work, which students could be expected to dislike.

On the other hand, one student outright stated, “…I don’t find SEURAT_Edu very useful. To me, it seems like in reality I am choosing my decisions regardless of what SEURAT_Edu says.” This type of thinking was evident in the experimental results where some students chose suboptimal data structures. This same student went on to say, “I am actually evaluating pros and cons, which I feel can be more easily done with pen and paper.” Though
a discouraging sentiment for documenting rationale, it makes sense considering this student’s previous statement. Much of the power of SEURAT_Edu comes in the form of recommendations based on user input. If this student is ignoring the recommendations and making choices regardless of them, then he really is missing out on much of what SEURAT_Edu offers.

6.3 CSE201 and CSE322

In spring of 2014, SEURAT_Edu was used inside two courses at Miami University. The first was CSE201, an introduction to software engineering course that is required by most of the computing-related majors at Miami University. As such, this course was comprised of mostly first and second year students as seen in Figure 26. For this assignment, students were asked to design a prototype conference room scheduling system.

![Figure 26 CSE201 Student Distribution](image)

The second course was CSE322, requirements engineering. This course provides a unique perspective from the previous two because it is geared toward third and fourth year students, as shown in Figure 27. Also, the content of this course lends itself very well to design rationale because it is focused around the discovery and representation of requirements.

![Figure 27 CSE322 Student Distribution](image)
Both of these classes used the same exact demonstration, assignment, and surveys. By using the same format for both of these courses, the results can be directly compared. Student were provided with six requirements for this assignment, as seen in Figure 28 below. Five of these were functional requirements, and the sixth was non-functional. Additionally, the instructor specified six design decisions that had to be addressed for the proposed system. Three of these decisions were provided with alternatives, including one that also received arguments for and against the alternatives. An alternative was selected for this last decision. When creating this assignment, the instructor opted not to complete a full solution version. The instructor and student version are the exact same.

![Rationale Tree](image)

Figure 28 CSE201 Starting Rationale
This assignment was designed to expose students to varying levels of Bloom’s Taxonomy as described in Table 1. These levels are displayed in this assignment as follows:

- **Knowledge:** Essentially regurgitating information, this level can be seen in the first decision, “Which requirement should be implemented first?” All applicable alternatives can be copied from the pre-defined list of requirements.

- **Comprehension:** This level is characterized by the students differentiating between given alternatives. The decision “Data structure to store schedule?” has three valid alternatives provided that the students must create arguments about.

- **Application:** Given a decision, students should be able to provide a list of alternative solutions. This is evident in several decisions, such as “Is authentication required?”

- **Analysis:** Students have to generate alternatives and then relate those alternatives back to the requirements as seen in “How should the user specify meeting start and end times?”

By creating such a range of challenges for the students, this gives the instructor the unique opportunity of seeing where each student lies within Bloom’s Taxonomy. This information can then be used to create assignments aimed toward the appropriate level for the class.

### 6.3.1 Demonstration

When creating a demonstration for these classes, several things had to be considered. First off, because CSE201 is placed so early in the course curriculum, many students are not familiar enough with data structures to compare them very well. For this reason, the previously used demonstration, creating a calendar, would not be very effective. Secondly, the class is required for many majors outside of computer science that do not require in depth programming knowledge.

Taking these into consideration, it was decided to not use a technical demonstration when first introducing SEURAT_Edu. Instead, students would be walked through an assignment where they were tasked with planning a party. Contrary to the previous demonstration, this one did not rely on any student input due to the potential of unforeseen suggestions. Figure 29 shows the starting rationale tree on the left, and the completed tree on the right. As shown below, these are decisions that most students are familiar with. And most importantly, the students were very engaged because of the light nature of the demonstration.
6.3.2 CSE201 Analysis

This analysis will mainly focus on three of the six decisions the students are faced with: which requirement to implement first, how to store the conference rooms, which data structure to use, and how to specify start/end times. The other decisions were too straightforward and resulted in very similar rationale among students.

The students’ first task was deciding which requirement to implement first. Because the instructor provided 5 functional requirements, there should have been at least 5 alternatives here, plus any extra from additional requirements the students created. Surprisingly, though, the second years are the only group of students that average more than 3.5 alternatives for this decision, as seen in Figure 30.

![Figure 30 CSE201 Decision 1 Alts](image)

What this means is that the students are most likely omitting the requirements that they think are incorrect, which should never happen. Students should always provide every valid
alternative they can come up with. This is even more unsettling because of the alternatives the students chose, shown in Figure 31. The most commonly selected alternative was adding a conference room to the system, which was one of the worst possible answers. For a prototyping system, it would be much more beneficial to get some of the core functionality working like scheduling a meeting or viewing the meeting schedules. Instead of providing functionality for adding conference rooms, it would be much better to hardcode a few into the system in order to move onto more difficult aspects of the system.

Looking at the students’ rationale, it seems like there were two common reasons for choosing to implement adding a conference room before anything else. First, many students wanted to implement everything in the order of when they would be used. For example, when using the system, the student would want to add a conference room, then create a meeting in the conference room, and finally view the meeting. The second reason for selecting this alternative was often that it would be the easiest to code. Many students created arguments about how easy it would be and provided very high amount values close to 10, while the other arguments’ values hovered around 5.

The next big decision the students were faced with was to decide on a data structure for storing the schedules. The instructor provided linked list, arraylist, and hashmap as three possible alternatives. No student added any additional alternatives for this decision. As seen in Figure 32, there was a good mix of every data structure being selected. Once again, this is a decision that seemed to have a clear best alternative in hashmaps. Upon inspecting students’ arguments, though, it seems that many students are simply not familiar with them. Many arguments were created against hashmaps because they are hard to use or “require advanced coding ability.” This is simply a result of how early in the computer science program most of these students are. Many have not yet taken a data structures course, which is when hashmaps are first introduced.

Figure 31 CSE201 Decision 1 Selected Alts
The final decision being considered for this assignment is how to specify start and end times for a meeting. This is the most complicated decision of this assignment due to how open ended it is. The students are required to interpret the decision and what it is asking in reference to the assignment, generate valid alternatives, and relate them back to the requirements. Depending on how the alternatives are interpreted, about a fourth of the students suggested having the user type the times into a text box while the other three fourths suggested presenting the user with available times. This is another situation where simplicity to code is just outweighing every other negative argument for the inferior alternative, which in this case is having the user type in the text box. It would be interesting to see if the students are setting the argument amounts before receiving a recommendation or after receiving a warning. The average number of alternatives provided per grade level are shown in Figure 33. The low number of alternatives is most likely a result of the students struggling to generate many options.

### Figure 32 CSE201 Decision 2 Selected Alts

![Data Structure to Store Schedule](image)

### Figure 33 CSE201 Decision 3 Alts

![Alts for 'Start/End Times'](image)

### 6.3.3 CSE322 Analysis

Just as in CSE201, only 3 decisions will be considered here: which requirement to implement first, what data structure to use to store the schedule, and how to specify start and end times for meetings.
When deciding which requirement to complete first, the CSE322 students should have had at least five alternatives, corresponding to the five provided requirements. As shown in Figure 34, this was proven true since each grade level at least five alternatives. This means that students in this course are considering every alternative as a possible solution and they are using SEURAT_Edu to help them make their decisions.

![Figure 34 CSE322 Decision 1 Alts](image)

Despite this, though, a large portion of the students still chose adding a room as the first requirement to implement. In general, the CSE322 students provided the same arguments as the CSE201 students: that it would be easier and that it is required first. However, several students provided four or more arguments for each alternative and still made that choice. In these instances where each alternative has many arguments for and against it, it is unlikely that the student simply increased a random argument’s amount in order to fit what they want. The distribution of what to implement first can be seen below in Figure 35.

![Figure 35 CSE322 Decision 1 Selected Alts](image)

The next decision the students faced was in determining what data structure would be best for storing the schedule in the prototype. Figure 36 shows how many students from each grade selected each alternative. There was a perfect split among all 3 data structures, but the distribution is skewed. The younger students favored arraylists while the older students favored hashmaps. By the time the students are taking CSE322, they should have taken the data structures course that teaches hashmaps and linked lists. What most likely happened is
that the younger students skipped data structures before taking this course, while the older students did not. This could be considered not a big deal since the prerequisites do not require data structures, but it would be interesting to figure out how this affects the students. There is potential for other assignments to require knowledge of hashmaps in order to effectively analyze requirements in this requirements engineering course. In these cases, the offending students would not be able to complete the assignments.

![Data Structure to Store Schedule](image)

Figure 36 CSE322 Decision 2 Selected Alts

Finally is the decision of how to specify start and end times for meetings. The majority of students opted for some method of letting the user select from available times, using dropdowns or a similar method. Only two students from CSE322 decided to go with a text box with manual time entry. The average number of alternatives created for this decision were very low, as seen in Figure 37. A possible explanation for this is that students already had an idea of how they wanted to address this decision and added an extra alternative or two to make it look like they were using SEURAT_Edu. However, another possibility is that the students struggled with such an open ended decision and had a hard time coming up with alternatives.

![Alts for 'Start/End Times'](image)

Figure 37 CSE322 Decision 3 Alts

6.3.4 Analysis Comparison

The benefit of using this same assignment in two separate classes is that it allows for a direct comparison of results. Each comparison was tested for statistical significance using
the Mann-Whitney U Test, using the hypothesis that CSE322 students made better decisions than CSE201 students. When comparing alternatives to determine who made better decisions, the alternatives were categorized as “good” or “bad” and each category was assigned a value. When looking at these two courses side by side, the results are somewhat similar, but there are enough differences to provide meaningful insight.

This can be seen as early as the first decision where students chose which requirement to implement first. Figure 38 shows the average number of alternatives created by students in each class. The spike in second year students from CSE201 is interesting, but the rest of the data points are about as expected. This shows that the CSE322 students have learned to consider all their possible alternatives from the beginning of the assignment instead of just omitting the ones that seem incorrect. These results are not statistically significant (U = 288.5, n₁ = 24, n₂ = 21, P = .41 > .05).

![Figure 38 Decision 1 Alternatives Comparison](image)

Despite the fact that the CSE322 students averaged more alternatives, their selections were much more concentrated. Figure 39 below represents the average number of students that selected each requirement to implement first. While the CSE201 students were all over the board with the highest portion of students selecting to add a room first, the CSE322 students were much less spread out. On top of this, a plurality of the CSE322 students chose to schedule a meeting first, which was the most correct answer. These results are not statistically significant (U = 327, n₁ = 24, n₂ = 21, P = .09 > .05).
When it came to choosing a data structure to store schedules, the overall results from each class showed much more resemblance. Figure 40 shows a side by side comparison of what portion of students from each class selected each alternative. However, it is important to remember that within CSE322 the youngest students accounted for most of the arraylist and linked list selections, while the older students selected hashmaps. These results are not statistically significant ($U = 283.5, n_1 = 24, n_2 = 21, P = .48 > .05$).

The final decision of how to specify start and end times was nearly identical between the two classes. Within each course, an overwhelming majority of students opted to allow the user to select, by varying methods, from available times. The remaining few students from each class opted for manual text entry. Across these selections there was no correlation by student year of study.

Overall, the results from CSE201 and CSE322 were pretty similar, despite some differences in the first two decisions. That being said, the CSE322 students tended to provide more alternatives and arguments as seen in Table 3. Similarly, fewer of the CSE322 students made decisions that were mainly influenced by ease of implementation. For these reasons, the CSE322 students provided better rationale than the CSE201 students. The
average number of decisions and average number of alternatives were both not statistically significant (U = 289, n₁ = 24, n₂ = 21, P = .41 > .05 and U = 305.5, n₁ = 24, n₂ = 21, P = .23 > .05 respectively). The average number of arguments, however, is significantly different (U=380, n₁ = 24, n₂ = 21, P = .0037 < .05).

Table 3 Assignment Comparison

<table>
<thead>
<tr>
<th></th>
<th>Average # Decisions</th>
<th>Average # Alternatives</th>
<th>Average # Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE201</td>
<td>7.1923</td>
<td>2.4972</td>
<td>1.7962</td>
</tr>
<tr>
<td>CSE322</td>
<td>7.5238</td>
<td>2.6346</td>
<td>1.9307</td>
</tr>
</tbody>
</table>

6.3.5 CSE201 Survey

Now that the usability of SEURAT_Edu had been assessed and the feedback had been incorporated, the survey was drastically revised and can be seen in Appendix E. As is evident from the survey, the main goals were to assess the impact of rationale documentation in general and to assess how SEURAT_Edu was received by the students. In order to determine how helpful the documentation process was, the students were asked if using SEURAT_Edu helped them generate more alternatives and if it encouraged them to put more thought into their decision making. Figure 41 shows that 54% of students considered more alternatives as a result of using SEURAT_Edu.

Figure 41 CSE201 More Alternatives

The distribution shown in Figure 41 represents the aggregate responses of students from each year of study. It is worth noting that a similar distribution occurs within each separate year of study.

When asked whether or not SEURAT_Edu caused students to put more thought into their decision making, the responses were a resounding yes. Figure 42 shows that 80% of students put more consideration into their decisions when using SEURAT_Edu. Once again, this distribution was consistent across each year of study.
Digging into the responses of those students that disagreed with this question proved interesting. Two of the three students that disagreed here left a comment stating that using SEURAT_Edu was too slow and acted as a “hinder to time management.” After reading these comments, one may expect that these students would not spend much time with the tool. However, all three students that disagreed provided more alternatives and arguments than the average student. For several decisions, these students even provided the most alternatives or arguments. One possible explanation is that these students did spend a significant amount of time completing this assignment, but do not realize the benefits. It would be interesting to have them do this same assignment again without documenting rationale to see how their decision making was different.

The next few survey questions relate directly to SEURAT_Edu and how useful the students found it. Figure 43 shows how many students would SEURAT_Edu again in the future for in-class assignments and personal projects.

Many students seem to be unsure about whether or not they found it useful enough to use it again. This is a question that would be interesting to ask the students about once they have had time to better familiarize themselves with the SEURAT_Edu. In regard to how many more students would rather use it for a personal project, the most common comment is that in-class assignments are not large enough to warrant the upfront time cost. Being that these
students are most likely still taking their introductory courses, this is a reasonable concern. Another student, however, stated that they would likely use it for personal decisions unrelated to programming.

Overall, these results prove encouraging for SEURAT_Edu. Most students seemed to notice an immediate impact on their decision making process after just one class period with the tool.

6.3.6 CSE322 Survey

The survey for this class is the exact same as the survey for CSE201 and can be found in Appendix E. Once again, the main goals of this survey were to assess the impact of rationale documentation in general and to assess how SEURAT_Edu was received by the students. To this end, students were asked how using SEURAT_Edu impacted them during the assignment and whether they would want to use it again in the future. Figure 44 represents the percentage of students that felt that documenting their design rationale helped them consider more alternatives.

![Figure 44 CSE322 More Alternatives](image)

A majority of the students in this course agreed that SEURAT_Edu helped them consider more alternatives. One student even stated, “I felt the need to argue for alternatives that I didn’t agree with at first in order to fill out more of the tree.” This statement epitomizes the need for documenting rationale. If not for using SEURAT_Edu, the student would have stopped considering new alternatives as soon as they found one that worked, even if it was suboptimal.

The next question was whether or not SEURAT_Edu helped the students put more thought into their decision making. Figure 45 displays the responses.
With 90% of students putting more thought into their decisions when documenting rationale, it seems safe to say that SEURAT_Edu was effective for this assignment. One student commented that, “[SEURAT_Edu] forced us to do decision making rather than just randomly picking decisions on the fly.” This testimony directly answers the research question of, “Does rationale documentation positively impact the decision making process?” There were no negative comments about SEURAT_Edu or rationale documentation in general here.

When asked if they would use SEURAT_Edu again in the future, the results are very inconclusive. As can be seen in Figure 46, the students are pretty evenly split between agreeing, disagreeing, and being undecided. The comments against using SEURAT_Edu in the future are the same that have been seen before. Students either do not want to take the time to document their rationale, or they just do not have projects that they deem large enough to justify using the tool.

6.3.7 Survey Comparison

This section looks at the differences between the survey results from each class. This is accomplished by looking at the percentage of students that selected each answer from each class. The Mann-Whitney U Test was used to determine the statistical significance of the hypothesis that CSE322 students provided more supportive responses than CSE201 students.
The “Strongly Agree” and “Agree” responses were combined and the “Strongly Disagree” and “Disagree” responses were combined in order to create more consolidated data.

Figure 47 represents whether or not students felt that using SEURAT_Edu helped them consider more alternatives. In each class, a plurality of the students agreed that it did in fact help them consider more alternatives. However, the CSE201 students provided a slightly less positive response to this question as shown by how many students disagreed. These results are not statistically significant (U = 264, n1 = 24, n2 = 19, P = .38 > .05). This difference in opinion is even supported by the data in Table 3, which shows that the CSE322 students averaged more alternatives than the CSE201 students.

![Considered More Alternatives](image)

**Figure 47** More Alternatives Comparison

Next the students were asked whether or not SEURAT_Edu helped them put more thought into their decision making. As seen in Figure 48, both classes provided overwhelmingly positive responses, with just a handful of students being undecided or disagreeing. Once again, the CSE322 responses were just slightly more positive and no one from the class strongly disagreed that SEURAT_Edu encouraged putting more thought into their decisions. These results are not statistically significant (U =252, n1 = 24, n2 = 19, P = .57 > .05).
Finally, the students were asked if they would use SEURAT_Edu again in the future for either class assignments or personal projects. For both questions, no students felt strongly that they would use SEURAT_Edu again. Several students did feel strongly against using it again, though. Though a large portion of students were undecided, this is not entirely discouraging. It can be expected that most students would not be in favor of documenting rationale due to the fact that it is additional work at the beginning of an assignment or projects. One way to interpret the large portion of students that answered undecidedly is that they saw the benefit but were deterred by the extra work. It would have been interesting to ask the students how they felt again after becoming more familiar with SEURAT_Edu. Another interesting observation from this data is that in both classes significantly more students would use SEURAT_Edu for personal projects than for class assignments. The differences between the CSE322 and CSE201 in both cases are not statistically significant ($U = 216$, $n_1 = 24$, $n_2 = 18$, $P = .99 > .05$ and $U = 210$, $n_1 = 24$, $n_2 = 18$, $P = .98 > .05$).

![Figure 48 More Thought Comparison](image1)

![Figure 49 Use SEURAT_Edu Again Comparison](image2)
Overall, the responses received by both classes were rather similar. The biggest noticed difference is that the CSE322 students reported slightly greater benefits from using SEURAT_Edu. That being said, both classes provided encouraging results that pointed toward noticed benefits of documenting rationale using SEURAT_Edu. In terms of using SEURAT_Edu again, however, the results of both classes were similarly inconclusive.

7 Future Work

The future of this research has two focuses: SEURAT_Edu development and experimentation. Fortunately, both of these can be done in tandem since they drive each other.

7.1 SEURAT_Edu Development

SEURAT_Edu is currently in a working state, but it could always be improved. Driven by the in-class experiments, it has improved dramatically since its first use in CSE274, but not every update has made its way into the tool yet. Below are outlined two specific areas that require significant development before group assignments can be used. More generally, the SEURAT_Edu should be updated incrementally as student feedback is received.

7.1.1 Roster

In order to create group assignments, the instructor needs an up-to-date roster that can be used to create student groups. The original idea was to retrieve the roster from Niihka in the same way that the user information is retrieved. However, this functionality is not yet available for external tools within the current version, 2.9, of Sakai at Miami University. This is the reason for the clunky roster creation interface currently implemented.

7.1.2 Group Assignments

Though support for group assignments has been included in the current release of SEURAT_Edu, it has not been sufficiently tested for use in the classroom. Also, concurrent access is not yet supported within the current group assignment functionality.

7.2 Experiments

Now that SEURAT_Edu has been developed, the research can continue on with a greater focus on the experiments. Many of the interesting questions of this research related to how rationale documentation effects student learning were never addressed due to the short time frame for experiments. This section outlines several different types of experiments that could be done with SEURAT_Edu.

7.2.1 Comparison Experiment

One of the most straightforward experiments to do would be a side-by-side comparison between a student not documenting rationale and a student that is. This could take place over the course of a single assignment or over the course of an entire semester. Given two equal students that perform consistently, this could go a long way in proving or disproving the effectiveness of documenting rationale on decision making and cognitive development.
7.2.2 **Nontechnical Experiment**

When designing SEURAT_Edu, one of the goals was to make a system that students from any field of study could utilize. However, only students registered for computer science courses have ever used it so far. It would be beneficial for SEURAT_Edu to get into the hands of more diverse users across different domains.

7.2.3 **Extended Experiment**

One of the main focuses of this research is to determine what kind of effects documenting design rationale has on cognitive development. However, because cognitive development is a process that students undergo throughout their entire college careers, this cannot be measured with independent experiments. This would require a long term commitment from student volunteers that would be willing to work with SEURAT_Edu consistently over an extended period of time.

8 **Conclusion**

The questions that drove this research were:

- Is there a relationship between rationale documentation and cognitive development?
- Can rationale be used to identify gaps in student knowledge?
- Does rationale documentation positively impact the decision making process?

So far, two of these questions have been addressed through the use of SEURAT_Edu in the classroom. First, SEURAT_Edu has proven promising in helping students during the decision making process. Just within the first few experiments run using SEURAT_Edu, students have begun to provide encouraging feedback, explicitly stating how helpful the tool is. In conjunction with the supportive survey results, SEURAT_Edu has already begun making a case for itself.

Through the use of SEURAT_Edu in the two conference room prototyping assignments, it has also already proved its worth in identifying gaps in student knowledge. The CSE201 instructor expected students to be familiar with hashmaps from a prerequisite course, but learned that it was no longer being taught in the course. Similarly, students in the CSE322 class would be expected to be familiar with hashmaps due to it being a 300 level course, but many of the students seemed very uncomfortable with them. Both of these occurrences could point toward the idea that the computer science curriculum does not spend enough time on hashmaps if students are getting by without being comfortable enough to use them.

The final question, determining whether or not there is a relationship between rationale documentation and cognitive development, has not yet been answered by this research. As suggested as future work, this will most likely require an extended study of a specific group of volunteers. However, with SEURAT_Edu developed and several preliminary in-class experiments already completed, this question could be addressed sooner than one might think.
9 Works Cited


Appendices

Appendix A SEURAT_Edu MySQL Tables

```sql
CREATE TABLE `requirements` (
    `id` int(11) AUTO_INCREMENT,
    `assignment_id` int(11) DEFAULT NULL,
    `turnin_id` int(11),
    `parenttype` int(11) DEFAULT '-1',
    `parentid` int(11) DEFAULT '-1',
    `name` text,
    `description` text,
    `type` int(11),
    `status` int(11),
    `importance` int(11),
    `enabled` tinyint(1) DEFAULT NULL,
    PRIMARY KEY (`id`)
)
```

```sql
CREATE TABLE `decisions` (
    `id` int(11) AUTO_INCREMENT,
    `assignment_id` int(11) DEFAULT NULL,
    `turnin_id` int(11),
    `parenttype` int(11) DEFAULT '-1',
    `parentid` int(11) DEFAULT '-1',
    `name` text,
    `designer` text,
    `description` text,
    `type` int(11),
    `status` int(11),
    `subdecision_required` tinyint(1),
    PRIMARY KEY (`id`)
)
```

```sql
CREATE TABLE `alternatives` (
    `id` int(11) AUTO_INCREMENT,
    `assignment_id` int(11) DEFAULT NULL,
    `turnin_id` int(11),
    `parenttype` int(11) DEFAULT '-1',
    `parentid` int(11) DEFAULT '-1',
    `name` text,
    `designer` text,
    `description` text,
    `status` int(11),
    `score` double DEFAULT NULL,
    PRIMARY KEY (`id`)
)
```

```sql
CREATE TABLE `arguments` (
    `id` int(11) AUTO_INCREMENT,
    `assignment_id` int(11) DEFAULT NULL,
    `turnin_id` int(11),
    `parenttype` int(11) DEFAULT '-1',
    `parentid` int(11) DEFAULT '-1',
    `name` text,
    `designer` text,
    `description` text,
    `type` int(11),
    `amount` int(11),
    `importance` int(11),
    `plausibility` int(11),
    `argument_type` int(11),
    `arguestype` int(11) DEFAULT NULL,
    `argues` text,
    PRIMARY KEY (`id`)
)
```

```sql
CREATE TABLE `claims` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `assignment_id` int(11) DEFAULT NULL,
    `turnin_id` int(11),
    `parenttype` int(11) DEFAULT '-1',
    `parentid` int(11) DEFAULT '-1',
    `name` text,
    `description` text,
    `importance` int(11),
    PRIMARY KEY (`id`)
)
```

```sql
CREATE TABLE `assumptions` (
    `id` int(11) AUTO_INCREMENT,
    `assignment_id` int(11) DEFAULT NULL,
    `turnin_id` int(11),
    `parenttype` int(11) DEFAULT '-1',
    `parentid` int(11) DEFAULT '-1',
    `name` text,
    `description` text,
    `importance` int(11),
    `enabled` tinyint(11),
    PRIMARY KEY (`id`)
)
```
CREATE TABLE `assignment_descriptions` (  
`id` int(11) AUTO_INCREMENT,  
`courseid` text,  
`title` text,  
`description` text,  
`duedate` date,  
`isAvailable` tinyint(1) DEFAULT NULL,  
`grouptype` int(11),  
`clustername` text,  
`hasStudentVersion` tinyint(1) DEFAULT NULL,  
PRIMARY KEY (`id`)  
)  

CREATE TABLE `assignment_turnins` (  
`id` int(11) AUTO_INCREMENT,  
`assignment_id` int(11),  
`user` text,  
`username` text,  
`groupnum` int(11),  
PRIMARY KEY (`id`)  
)  

CREATE TABLE `statuses` (  
`id` int(11) AUTO_INCREMENT,  
`assignment_id` int(11),  
`turnin_id` int(11),  
`rat_type` int(11),  
`rat_id` int(11),  
`error_level` int(11) DEFAULT '0',  
`description` text,  
PRIMARY KEY (`id`)  
)  

CREATE TABLE `groups` (  
`courseid` text,  
`clustername` text,  
`groupnum` int(11) DEFAULT NULL,  
`user` text  
)  

CREATE TABLE `rosters` (  
`courseid` text,  
`user` text  
)
Appendix B Student Waiver

Rationale, Writing, and Critical Thinking, Miami University 2013

Investigators:

- Janet Burge. Principal Investigator. Miami University Computer Science and Software Engineering Department. Burgeje-at-miamioh.edu
- John Malloy. Graduate Assistant. Miami University Computer Science and Software Engineering Department. Malloyjw-at-miamioh.edu

Purpose and Benefits:

This project is designed to investigate if focused writing assignments and design rationale can help to encourage critical thinking, idea generation, and reflection. The results of this study will be submitted for publication (with information identifying the subjects removed) and will also be used to guide further educational research at Miami University. Identifying information associating students with assignments will not be maintained past the end of the semester.

Other Information:

Agreeing to allow your assignment to be used as part of this study is completely voluntary and will not affect your grade. You can choose to withdraw from the study at any time. Your instructor will not be aware of which students have consented to participate. Please indicate your agreement to participate in this study.

___ By checking this box I am verifying that I am at least 18 years of age. I am also aware that this study is voluntary. I am also aware that if I have questions about this research or wish to withdraw from the study I may contact Janet Burge, burgeje@miamioh.edu, 513-529-0347 or Thomas Barber, barberta@miamioh.edu, (812) 701-5044, and if I have questions about my rights as a participant in a research study I may contact the Institutional Review Board at Miami University, (513) 529-3734 or via email at (humansubjects@miamioh.edu).

I understand that the results of this experiment will be published and give my permission for this to happen under the condition that publication will be done anonymously and that personal information collected as part of the experiment will only be seen by the investigators.

Name: __________________________________________

Signature: __________________________________________ Date: _________
Appendix C Student Demographic Survey

Printed Name: ________________________________

Demographic Information

If you would prefer to not answer a question, leave it blank.

1. What is your age?

2. Which best classifies you?
   a. Undergraduate Student
   b. Graduate Student
   c. Working (or Retired) Professional

   (If you answered c to question 2, skip to question 6)

3. What year of study are you in?
   a. First Year
   b. Second Year
   c. Third Year
   d. Fourth Year
   e. Other:

4. Are you required to take this course to get a degree in your major?
   a. Yes
   b. No

5. What is your major?

6. What is your profession?

7. Have you had any previous training on the topic taught in this class?
   a. Yes
   b. No

8. What is your gender?
   a. Male
   b. Female
Appendix D Student Post-Test Survey (CSE274)
Post-Test Usability Questionnaire: Student Version

Please circle one answer for each question:

Usefulness and Effectiveness of SEURAT_Edu

I was given sufficient instructions and time to complete the tasks for this experiment.

Strongly Agree   Agree   Undecided   Disagree   Strongly Disagree

Why:

It was easy to start work on a new assignment using SEURAT_Edu.

Strongly Agree   Agree   Undecided   Disagree   Strongly Disagree

Why:

It was easy to enter requirements using SEURAT_Edu.

Strongly Agree   Agree   Undecided   Disagree   Strongly Disagree

Why:

It was easy to enter decisions using SEURAT_Edu.

Strongly Agree   Agree   Undecided   Disagree   Strongly Disagree

Why:

It was easy to enter alternatives using SEURAT_Edu.

Strongly Agree   Agree   Undecided   Disagree   Strongly Disagree

Why:

It was easy to enter arguments using SEURAT_Edu.

Strongly Agree   Agree   Undecided   Disagree   Strongly Disagree

Why:
It was easy to enter claims using SEURAT_Edu.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Why:

It was easy to enter assumptions using SEURAT_Edu.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Why:

I could easily see how each design alternative was evaluated by SEURAT_Edu.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Why:

I could easily change my decision priorities in SEURAT_Edu and see how the SEURAT_Edu recommendations change.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Why:

I can easily edit my rationale using SEURAT_Edu.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Why:

What Features of SEURAT_Edu did you find to be the most useful?

Were there features that you did not find to be useful or that require improvement before you would want to use this for a class assignment?
Appendix E Student Post-Test Survey (Revised)

Name:

Post-Test Usability Questionnaire: Student Version

Please circle one answer for each question:

Usefulness and Effectiveness of SEURAT_Edu

It was easy to start work on a new assignment using SEURAT_Edu.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Comment:

It was easy to create/modify pieces of rationale.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Comment:

I could easily see how each design alternative was evaluated by SEURAT_Edu.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Comment:

SEURAT_Edu helped me consider more alternatives than I may have considered without it.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Comment:

SEURAT_Edu helped me put more thought into my decision-making.

<table>
<thead>
<tr>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
</table>

Comment:
I found SEURAT_Edu to be useful for this assignment.

Strongly Agree    Agree    Undecided    Disagree    Strongly Disagree

Comment:

I would use SEURAT_Edu for future in-class assignments.

Strongly Agree    Agree    Undecided    Disagree    Strongly Disagree

Comment:

I would use SEURAT_Edu for future personal projects.

Strongly Agree    Agree    Undecided    Disagree    Strongly Disagree

Comment:

What Features of SEURAT_Edu did you find to be the most useful?

Were there features that you did not find to be useful or that require improvement before you would want to use this again?

Final comments: