SEURAT_Edu: A Tool to Assist and Assess Student Decision-Making in Design

John Malloy
Computer Science and Software Engineering
Miami University
Oxford, Ohio, USA
malloyjm@miamioh.edu

Janet Burge
Mathematics and Computer Science
Wesleyan University
Middletown, CT 06459
+1 860-685-2413
jburge@wesleyan.edu

ABSTRACT

Decision-making is central to software development. Most problems do not have straightforward solutions but instead require assessing and evaluating multiple candidate solutions depending on the requirements (functional and non-functional) for the specific project. Real problems may have more than one valid solution and software engineering students need to be able to synthesize design alternatives and reason about them to make the best decision for the specific problem under consideration. The work described here uses a web-based system to administer rationale-based assignments where the students are required to explicitly present their rationale for a series of decisions and use the capabilities of the tool to justify their choices. Students using the tool report that it helped them consider more alternatives and put more thought into their decision-making process.

Keywords

Design Rationale; Decision-Making; Learning Management Systems; Bloom's Taxonomy

1. INTRODUCTION

As software developers, students need to be able to take the concepts they've learned in earlier classes and apply them to larger, more realistic problems in later ones (and in the real world after graduation). This is not always as straightforward as it sounds—real problems may have more than one valid solution and students need to be able to understand these problems, propose candidate solutions, and analyze the solutions based on a set of criteria that they have to discover as part of the process.

Many students find this challenging. Students may have difficulty dealing with problems where there is more than one right answer. They may also have difficulty objectively looking at alternatives when their preferences are biased in favor of past experience or personal goals that may not mesh with those of their client (such as how easy the solution is to code—a high priority for students who have demands on their time beyond the one class but not

necessarily a priority for the client). This can be a problem if students fixate on a solution that may be less than optimal rather than explore new options.

One way to encourage generation and analysis of alternatives is to require that students create explicit argumentation documenting the decisions they need to make, alternatives considered, and the arguments for and against these alternatives. Argumentation-based design rationale is one mechanism for doing this. The rationale can be used to perform a simple numeric evaluation of the alternatives to encourage students to justify their choices.

We developed a web-based tool, SEURAT_Edu, to allow instructors to create rationale-based assignments suitable for students at different stages in the curriculum. The goal is to use these assignments to help them grow in their ability to identify, reason about, and select design alternatives. SEURAT_Edu integrates with existing Learning Management Systems (LMS) to make it easy for instructors to administer and evaluate student work.

This paper is structured as follows. Section 2 discusses related and prior work in educational research and using rationale in education; section 3 describes SEURAT_Edu, section 4 gives usability results from classroom use and section 5 provides a summary, conclusions, and future work.

2. RELATED AND PRIOR WORK

We designed SEURAT_Edu to help students become better designers by recording their rationale, something that has been shown to result in higher quality designs for inexperienced designers [1]. We observed that students found designing to be challenging and looked at existing models of student learning for help.

Two education models inspired this work. The first is Bloom's Taxonomy [2], a taxonomy of learning levels that is often used in formulating educational objectives. These levels are knowledge (memorizing concepts in a domain), comprehension (understanding these concepts), application (applying the facts to a new situation), analysis (reasoning about the concepts), synthesis (forming new concepts), and evaluation (assessing and judging using concepts and criteria). An argumentation-based approach appears to be a good fit for encouraging and assessing student ability to analyze and evaluate.

The second theory is the Perry scheme for Intellectual and Ethical Development [3]. Perry describes how students progress in development starting at dualism, where students believe there are always right or wrong answers, into multiplicity, where all options are equally valid, into relativism, and finally commitment.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SIGCSE'16, March 2–5, 2016, Memphis, TN, USA.
© 2016 ACM. ISBN 978-1-4503-3685-7/16/03...\$15.00.
DOI: <http://dx.doi.org/10.1145/2839509.2844555>

Moving through these stages requires the ability to assess alternatives and make choices when faced with changing criteria.

In addition to student progression, evaluating alternatives also plays a role in encouraging creativity. One of the key stages in the creative process is ideation—the generation of multiple ideas. Ray McCall [4] discusses ideation in his work, but he combines it with evaluation. 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.”

In our earlier work, we performed a preliminary exploration of using argumentation to encourage exploration of alternatives [5] by comparing results of two design methods: using the Eclipse-based SEURAT (Software Engineering Using RATIONale) plugin for rationale capture [6] and using Pugh Matrices [7] to capture alternatives and criteria. SEURAT provided the additional assistance of automatically assessing alternatives based on criteria while the Pugh Matrix was a more manual approach. The goal for using both tools was to encourage students to document their reasoning with the hope that this would cause them to delay making a decision until more analysis had occurred. Results showed that students using rationale-based approaches did not consider more alternatives (they were given a target number by the instructor) but these approaches did encourage reflection and were useful tools for faculty to identify where some of the difficulties were. This experiment only asked the students to provide rationale for a single decision.

Tang, et al. [1] conducted a study where participants were asked to apply design reasoning by capturing rationale both as they were designing and when prompted by interviewers. The designers in the experimental (rationale) group had more awareness of their own reasoning, considered usability throughout the process, were more likely to shift their initial design impression as they gained in knowledge, and did more backtracking to re-investigate previous decisions. The impact of using design reasoning was more apparent for less experienced participants (5 years of experience or less) than it was for more experienced designers.

Van Heersh, Avgeriou, and Tang [8] investigated the use of decision viewpoints as a way to encourage junior designers to investigate solution alternatives. They performed an experiment using four groups of software engineering students towards the end of their undergraduate education. They used three decision viewpoints in their study—a decision detail viewpoint, which gives information about architectural decisions that includes their rationale, a decision relationship viewpoint, that would include information about which decisions depend on (or are alternatives to) each other, and a decision forces viewpoint which gives the relationship between the decision and criteria (forces) that need to be considered when making architectural choices. They found that the latter viewpoints (forces and relationships) helped the students explore and evaluate design options and also helped them assess the results of combining different design options. The students did not like documenting the decision detail viewpoint and did not consider it useful for their own work but did say that they felt it would be helpful for subsequent developers who may take over the project in the future.

3. SEURAT_Edu

Our earlier work [5] used SEURAT to capture and evaluate rationale. SEURAT was designed to support software development and maintenance, not as a tool for assignment

administration. This meant that using SEURAT had had a number of disadvantages.

1. *Eclipse Specific* – using SEURAT means using Eclipse. Students taking CS classes that are not using Eclipse are confused by the need to switch tools for the rationale-based assignments. We also want to eventually expand our experiments beyond computer science since decision-making is required in many other fields. Non-CS students and faculty would be unlikely to want to try the tool if it is designed only for software developers.
2. *Overly Complex* – SEURAT includes features designed to support software development that are not necessary for the classroom setting. A simplified version of the tool is easier for students and faculty to learn.
3. *Limited Access* – SEURAT uses a database local to the machine that Eclipse is installed on to store the rationale. This means that a student could only work on their rationale on a single computer in a computer laboratory with controlled access. This limited rationale use to small exercises and did not support capturing rationale as part of homework assignments.
4. *No Administration* – SEURAT is not set up to deliver pre-generated rationale to the students or to deliver rationale to faculty to be graded. Database expertise is required to ensure that students have access to rationale already in the database, either by installing an already populated database locally on each machine or by creating duplicate copies of the rationale in a shared database. There is no way to enforce deadlines with the tool. Analyzing student work is a challenge using SEURAT since the student database tables have to be accessible to the instructor. The instructor has to use SQL queries or to load the student data as a new SEURAT project in order to assess the rationale.

To address these issues, we developed a new system: SEURAT_Edu. SEURAT_Edu is purely web-based where rationale is stored in a shared MySQL database. It integrates with a variety of Learning Management Systems (LMSs) using the Learning Tools Interoperability (LTI) framework. This means that authentication and assignment administration are taken care of by the LMS. This means students do not need to register with a second system in order to complete rationale-based assignments but instead can access them directly through the LMS already used for the course. It also means that instructors can easily view the rationale trees just like the students—making it easy to view the rationale.

Moving to a web-based environment provides a tool that is not CS-specific but could be used in any subject area where faculty want students to develop and practice their skills at identifying and evaluating alternatives. Complexity was addressed by only implementing a subset of SEURAT’s capabilities and the tool was designed to provide instructor support in developing and assessing assignments. The experiments described here were administered using a Sakai-based LMS (<https://sakaiproject.org>), but SEURAT_Edu has also been successfully integrated with Moodle.

3.1 Support for Assignment Creation and Administration

Rationale-based assignments can be tailored to the specific class needs and level of student by providing the students with starting rationale. SEURAT_Edu lets instructors create a set of “solution”

rationale and then use that as the basis of a student version that may have some of the content modified or removed, as appropriate. The instructor is not required to have a complete rationale solution unless they prefer to start with one. The assignment due-date is set within SEURAT_Edu. Once the due-date has passed the students can view their solutions but no longer modify them. The instructor also can control if the assignment is visible to the students or not from inside SEURAT_Edu. Instructors can use SEURAT_Edu to view student submissions. Rationale in SEURAT consists of the following elements:

- *Requirements* – the system requirements, functional or non-functional, for something the students are required to design or define as part of the assignment. SEURAT_Edu also supports entering rationale for the requirements.
- *Decisions* – decision problems that the students need to solve as part of the assignment. Each decision can have either alternatives or sub-decisions, but not both.
- *Alternatives* – options considered as possible solutions for the decision problems posed by the assignment. 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 we want to capture all options in the tool.
- *Arguments* – 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.
- *Claims* – criteria used in decision-making that were not explicitly given as requirements.
- *Assumptions* – reasons that the user believes to be true but is unsure about, or things that might change over time.

Figure 1 shows the types of rationale in their hierarchical structure.

Figure 1. Rationale Types

Rationale is visualized using a tree format with colored icons differentiating between the different rationale elements. Rationale is entered via a form-based interface with default values provided for all fields except for the name of the rationale element. As rationale is entered, it is displayed in the rationale tree. Figure 2 shows a sub-set of a rationale tree giving a decision (How to persistently store data in the system), two alternatives, and some arguments.

Figure 2. Rationale Tree in SEURAT_Edu

After creating the starter rationale, the instructor can then create a due-date for the assignment (the date after which the student will no longer be able to edit their rationale) and publish the rationale to the class through the LMS. The LMS uses the class roster to create copies of the rationale tree for each individual student so they can complete their assignments independently.

3.2 Support for Students Working on Assignments

The students use the starter rationale to complete the assignment designed by their instructor. They can view and edit the starter rationale and also add new elements. SEURAT_Edu encourages the students to use and complete the rationale by displaying its status. The rationale entered into SEURAT_Edu is evaluated by the system to look for errors in the rationale and/or errors in reasoning. These errors are shown in two places. One is in the rationale tree. There are small overlaid status icons to indicate where there are errors or warnings. One can be seen on the “How to persistently store data in the system” decision in Figure 2 – the small X (shown in red by the system) overlaying the lower left corner of the icon. The second place to see the results of the evaluation is in a Rationale Status Panel that lists all the errors and warnings. Figure 3 shows some initial errors given by SEURAT_Edu for the starting state of an assignment.

Status Panel

- ⚠ No alternative selected for decision 'Data structure to store schedule?'
- ⚠ No alternative selected for decision 'Which requirement should be implemented first?'
- ⚠ No alternative selected for decision 'How are conference rooms stored?'
- ⚠ No alternative selected for decision 'Authentication required to view?'
- ⚠ No alternative selected for decision 'How should the user specify meeting start and end times?'

Figure 3. Rationale Status Panel

The rationale status is used by the students to see if the rationale that they have entered in their solution is consistent and complete. The following types of errors are detected and reported by SEURAT_Edu:

- *Requirement violation* (error) – an alternative has been selected where one of the arguments against it indicates that the alternative violates a requirement.
- *No alternative selected* (error) – the student has not selected an alternative for a decision problem.
- *Decision has too many selected alternatives* (error) – some decisions can be marked as multi-selection where more than

one alternative can be selected but for many only one choice is allowed. If only one choice is allowed but more than one alternative is selected an error will be reported.

- *Alternative has no arguments in its favor* – if a student selects an alternative without describing any reasons why, SEURAT_Edu reports this as an error.
- *A selected alternative requires that another alternative also be selected but it was not* (error) – if there are dependencies on alternatives and only one is selected by the student SEURAT_Edu will point out the error.
- *Duplicate arguments found* (warning) – if the same argument appears twice for an alternative that could be an error or could indicate that arguments are not given at a fine enough level of detail (for example, there could be two reasons why an alternative is “usable,” in which case the specific reasons should be given rather than the more generic one appearing twice).
- *The highest rated alternative was not selected* (warning) – this appears if the alternative selected did not have as much support in the rationale as another alternative for the same decision. This means either a different choice should be made or more arguments for the current choice should be provided.

4. TOOL ASSESSMENT

SEURAT_Edu was used in three courses: a 200-level Data Structures (DS) course, a 200-level Software Engineering (SE) course and a 300-level Requirements Engineering (RE) course. The Data Structures course contained a laboratory exercise where the students used SEURAT_Edu to explore some design alternatives. This was an opportunity to test out the tool in a real educational setting and to dry-run our usability data collection. The survey instruments were re-written after this experiment because usability feedback on individual tool features turned out to be less interesting and useful than an assessment of the tool as a whole.

For the SE and RE classes, the same assignment was used in both so that responses could be compared. The RE course had the SE course as a pre-requisite so it was not possible to have a student taking both courses during the semester where the assessment was administered. Seventeen students participated from the SE class—ten second year students, four third year students, and three fourth year students. Nineteen students participated from the RE class—two second year students, seven third year students, seven fourth year students, and three fifth year students.

4.1 Assessment Structure

Each assignment had one in-class day (80 minutes) that consisted of four distinct parts:

- *Introduction/Waiver:* At the beginning of the class period, the instructor introduced the assignment and gave 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 demonstration consisted of walking the students through accessing the tool, viewing the assignment, and modifying rationale. The demonstration was conducted using a projector, while the students watched, asked questions, and made suggestions for rationale that could be entered for the demonstration assignment. The demonstration used party planning as an example rather than a technical problem to keep the focus on the tool rather than having students try to learn how to use the tool and how to understand the problem simultaneously.
- *Assignment:* After the demonstration, the students started the assignment in lab. 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. The assignment was structured this way to lessen the impact of learning effects on the rationale. We also did not want to penalize students who spent more time exploring the new tool rather than jumping right into completing the assignment.
- *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. The survey was given in class the day the assignment was given to ensure that only students who attended the demonstration were evaluating tool usability and to ensure that all students completed the survey. A disadvantage of this approach is that students were only making their judgments based on their initial in-class use.

4.2 Assignment

The students were given starter rationale for a conference room scheduling system. The assignment gave a brief description of the system, listing the five main user features (schedule meetings, delete meetings, view a weekly schedule for a room, and browse back and forth in time when viewing) as well as an administrative feature—the ability to add new conference rooms. The six features were listed as requirements in SEURAT_Edu. There were also three Claims given (easy to code, scalable, and quick access) and one Assumption (no need to store much data in the prototype).

The students were asked to complete the rationale for six decisions:

1. “How to persistently store data in the system?” The alternatives of “a RDBMS” or “object serialization” are given, along with fully specified rationale.
2. “How are conference rooms stored?” Two alternatives are given: “string giving the name” and “a conference room class”.
3. “Data structure to store schedule?” Three alternatives are listed: ArrayList, linked list, and HashMap. There are no arguments given for any of the alternatives.
4. “Which requirement should be implemented first?” For this decision, the students are not given any alternatives although they do have the list of requirements.
5. “How should the user provide start and end times?” No alternatives are given.

6. “Is authentication required?” No alternatives are given.

The Rationale Status Panel displays errors and warnings from the rationale. The students are asked to complete the rationale so that there are no more errors or warnings—this means that an alternative must be selected for each decision and the selected alternative needs to have the same or more support indicated as its competing alternatives.

The starter-rationale was deliberately provided at different levels of completion:

- Fully specified rationale where the student only needed to make a choice.
- Decisions where alternatives were specified but where the students would need to use their own knowledge to specify arguments for and against the alternatives.
- Decisions where the students would need to specify both the alternatives and the arguments.

4.3 Survey Results

The students were given a survey towards the end of the lab session to ask them about their experience using SEURAT_Edu. We chose to administer it in the same class where the assignment started to ensure that only the students who had received training on the tool filled out the survey. Questions asked for answers on a Likert Scale, where answers ranged from Strongly Agree (SA) to Strongly Disagree (SD). The survey questions asked about both the usability and utility of the tool.

For usability, the students were asked if they agreed with the statement “It was easy to create/modify pieces of rationale.” Responses from the two classes are shown in Figure 4. Most students agreed with that statement although six of the SE students were undecided and two from each class disagreed, although not strongly. One undecided student expressed tool usability concerns as did several of the disagreeing students. We do not know if the results might be different if the students had filled out this survey after completing the assignment rather than at the end of the class period where they were first introduced to the tool.

Figure 4. Easy to Create Rationale

Students were also asked if it was easy to see how the tool evaluated the alternatives (Figure 5). More students agreed than disagreed but for each class about a third were undecided. This could have been because they had only learned to use the tool that day. One SE undecided student said they would not have understood the tool without the demo and another did not understand how the numbers input into the argument were arrived at: “Some aspects were unclear. Why did I have to choose an

importance/plausibility for arguments? What do these values mean? How can they be measured?” One of the SE students who disagreed that it was easy to see the evaluations gave the following reason: “I had a little trouble figuring out how each decision was evaluated with what I put in, but that could just be inexperience with the system.”

Figure 5. Easy to See Evaluations

We also asked four questions about the utility of the tool. First, the students were asked if they agreed or disagreed with the statement “SEURAT_Edu helped me consider more alternatives than I may have considered without it” (Figure 6). Most students agreed although there were still students who were undecided and a few who disagreed (although only one strongly). None of the undecided students gave a reason but one of the disagreeing students commented that they had to manually enter the alternatives (which was the point of the assignment).

Figure 6. Helped Consider More Alternatives

The students were also asked if they agreed with the statement “SEURAT_Edu helped me put more thought into my decision-making” (Figure 7). Most students felt that it did. Some of the students who disagreed commented on the time required by the tool but it is interesting to note that the three SE students who disagreed all provided more alternatives and arguments than the average student, suggested that they put quite a bit of thought into their decision-making. The students were also asked to agree or disagree with the statement “I found SEURAT_Edu to be useful for this assignment” (Figure 8).

The majority of the RE students found the tool useful for the specific assignment and a little less than half of the SE students found the tool useful. The survey was given before the students had finished the assignment, which may have impacted the results. Two of the undecided SE students provided reasons—one

felt the assignment was not large enough to need the tool and the other felt the assignment itself was too confusing.

Figure 7. More Thought into Decision-Making

Figure 8. Usefulness for This Assignment

5. SUMMARY AND CONCLUSIONS

The results of the surveys suggest that most of the students did feel they considered more alternatives and made better decisions with the tool than they would have without the tool. The student survey results might have been different had the surveys been taken after the assignment was complete (when students would have been more familiar with the tool and the impact of just having learned to use it would be lessened) but we were concerned about students taking the survey who had not participated in training (all students were required to participate in the assignment but we only reported on results for those who were trained and who had consented to have their results used in the study). We suspect there would have been fewer undecided ratings if the students had more time with the tool although we can only speculate on the direction of those ratings.

We are currently extending SEURAT_Edu to allow group projects and to include analysis tools for use by the instructor and researchers. Analyzing the rationale requires either manual analysis of the student rationale or having direct access to the database to run SQL queries. Supporting common analysis queries will allow SEURAT_Edu to scale to larger classes to help instructors identify where students might have misconceptions about the material—such as not understanding some of the alternatives or how the criteria might apply. Our preliminary analysis highlighted some of these issues. For example, the

software engineering instructor did not realize the course prerequisites had new learning outcomes that no longer taught Java Collection classes like HashMap. This suggests an alternative use for SEURAT_Edu where students can have rationale-based assignments early in the course as way to determine where there might be gaps in prior knowledge that should be addressed at the start of the class.

We would also like to perform a longer study investigating the ability of the tool to help students progress in their reasoning abilities through the curriculum and to perform some controlled studies that assess performance on making design decisions assistance of with and without the tool.

While more work can and should be done to study the impact of using SEURAT_Edu, these initial experiments provided useful information on where there were gaps in student knowledge and where more time needed to be spent to address these issues. In particular, we uncovered a need to focus more on requirements prioritization. Using SEURAT_Edu to provide decision-centered assignments is a good start towards identifying where students may need more instruction and provide a springboard for discussion of these issues in class.

6. ACKNOWLEDGMENTS

This work was supported by NSF CAREER Award CCF-0844638 (Burge). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

7. REFERENCES

- [1] A. Tang, M. Tran, J. Han, H. vanVliet (2008). Design Reasoning Improves Software Design Quality, International Conference on the Quality of Software Architectures, pp. 28-42.
- [2] B.S. Bloom (Ed.). (1956). Taxonomy of educational objectives: The classification of educational goals. New York: David McKay Company, Inc, 1956.
- [3] W.G. Perry, Forms of intellectual and ethical development in the college years: A scheme. New York: Holt, Rinehart, and Winston, 1970
- [4] R. McCall, "Critical Conversations: Feedback as a Stimulus to Creativity in Software Design." Human-Computer Interaction. 2013. Volume 20. 11-40.
- [5] J. Burge, B. Brinkman, Using Rationale to Assist Student Cognitive and Intellectual Development, in Human Technology: An Interdisciplinary Journal on Humans in ICT Environments, Volume 6 (1), May 2010, 106–128
- [6] J. Burge, D.C. Brown, "An Integrated Approach for Software Design Checking Using Rationale", Design Computing and Cognition '04, J. Gero (Ed.), Kluwer Academic Publishers, Netherlands, 2004, pp. 557-576
- [7] S. Pugh, *Total design*. Wokingham, UK: Addison-Wesley Publishing Company, 1991.
- [8] U. Van Heesch, P. Avgeriou, A. Tang, "Does decision documentation help junior designers rationalize their decisions? A comparative multiple-case study", Journal of Systems and Software, 86, 2013, pp. 1545-1565