ECoDE, Ectropic Collaborative Design Environment

A Study in Helping Students Learn Object Oriented Software Design

by
Kathleen Arnold Gray

College of Computing
Georgia Institute of Technology

Abstract

One distinct benefit that Open Source software development offers is the expediency of allowing many developers to contribute to a particular program without necessarily having specific knowledge of the overall program. These developers may simply want to adapt a program to a specific problem but their individual fixes and enhancements may be beneficial to the overall purpose of the program. Open Source proponents envision each developer contributing to the same code base. However, two limitations exist to this approach: the inherent nature of all evolving software to lose its design coherence over time and the need for a central individual to coordinate the evolution of the software.

Ectropic software is an attempt to leverage the advantages of Open Source while overcoming the limitations. Ectropy is the inverse of entropy. Ectropic Design is a design method by which order and structure are created out the efforts of multiple, unrelated software developers. A combination of Ectropic Design and an active collaboration space designed for support, allows Ectropic software to evolve, becoming more highly structured and specifically able to accomplish the goals of its users, without the need of a central human coordinator.

In the design of ECoDE, Ectropic Collaborative Design Environment, we have captured two key components of Ectropic Design: Scenario-based Analysis and Collaborator (CRC) Cards. This study is specifically focused on testing ECoDE as an Ectropic Design environment for usability and as an educational aid in teaching software design.

Introduction

Description of the Problem

One distinct benefit that Open Source software development offers is the expediency of allowing many developers to contribute to a particular program without necessarily having specific knowledge of the overall program. These developers may simply want to adapt a program to a specific problem but their individual fixes and enhancements may be beneficial to the overall purpose of the program. Open Source proponents envision each developer contributing to the same code base. However, two limitations exist to this approach: the inherent nature of all evolving software to lose its design coherence over time and the need for a central individual to coordinate the evolution of the software.

Ectropic software is an attempt to leverage the advantages of Open Source while overcoming the limitations. Ectropy is the inverse of entropy. Ectropic Design is a design method by which order and structure are created out the efforts of multiple, unrelated software developers. It is feature-oriented design. Software evolves ectropically through the continuous augmentation of its features, which are bound to specific program goals. These evolving

features are defined in terms of the end-user goals they achieve and how the features interact, both statistically and dynamically, with other features.[1]

Ectropic Design facilitates the development of software by multiple, unrelated developers working concurrently. By binding source code and collaboration technology to specific program goals, Ectropic Design provides developers with the necessary mechanisms to enhance software continuously, while maintaining the conceptual integrity of the program.

This study is focused on testing ECoDE, Ectropic Collaborative Design Environment, for usability and as an educational aid in teaching software design.

Description of ECoDE

Ectropic Collaborative Design Environment, ECoDE, is a development tool designed to capture two key components of Ectropic design: Collaborator (CRC Cards) and Scenarios. ECoDE includes a graphical user interface targeting novice software designers and attempts to present an environment that couples the flexible and modular structure of well-designed object-oriented software and perspicuity of functionally organized software.

The main interface is called an Ectropic Design Navigator and captures two main tools: The CRC Card Navigator and the Scenarios Navigator.

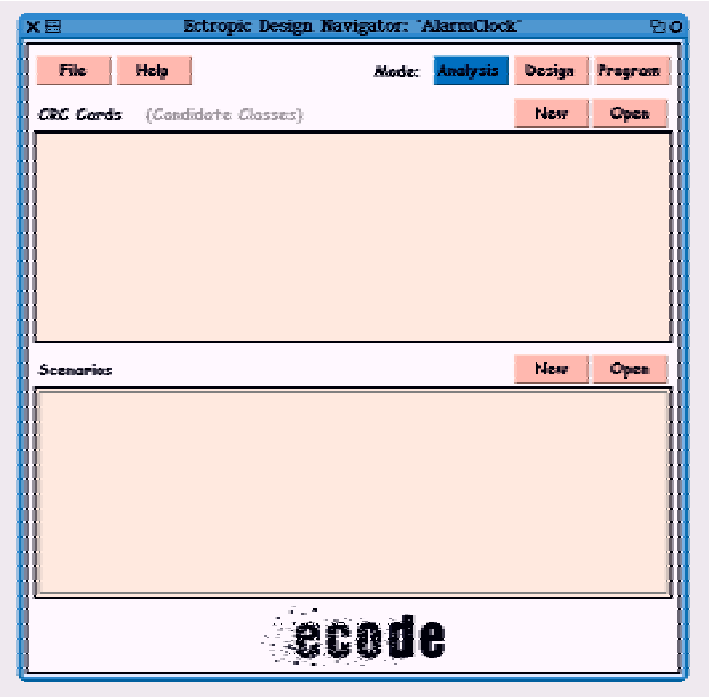


Figure 1. ECoDE's Ectropic Design Navigator.

ECoDE divides the object-oriented design process into three distinct phases, or *modes*:

- Analysis Mode – This mode allows the user to identify and analyze program goals. Additional tasks include identifying candidate classes (CRC Cards) to meet the goals, determining the services or responsibilities required and distributing these responsibilities among the candidates. ECoDE provides the designer with the ability to:

  - Create CRC Cards that will collaborate to meet the program goals.

- Create Scenarios for identifying the required Responsibilities
- Build Scenarios from a sequence of CRC Card – Responsibility pairs (called *Episodes*)
- Assign Responsibilities to CRC Cards

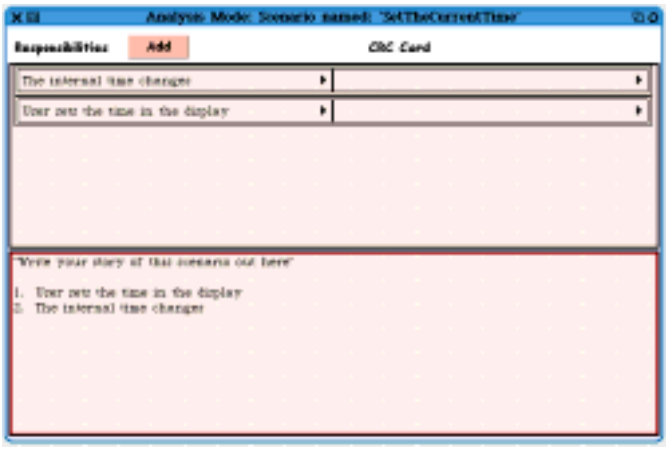


Figure 2a. A Scenario Analysis.

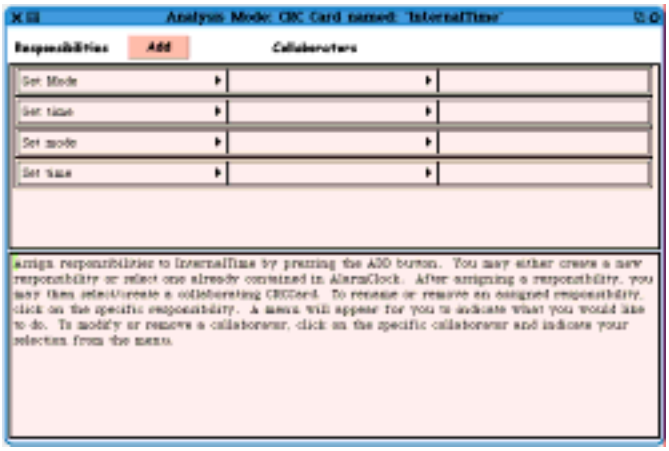


Figure 2b. A CRC Card Analysis.

- Design Mode – The objective during the Design mode to design an implementable and complete design. Specifically, each CRC Card should be reviewed and for each responsibility assigned to the card, a corresponding method should be assigned. ECoDE provides the designer with the ability to create, categorize, and describe methods and match them with specific responsibilities, in addition to all tasks available in Analysis mode.

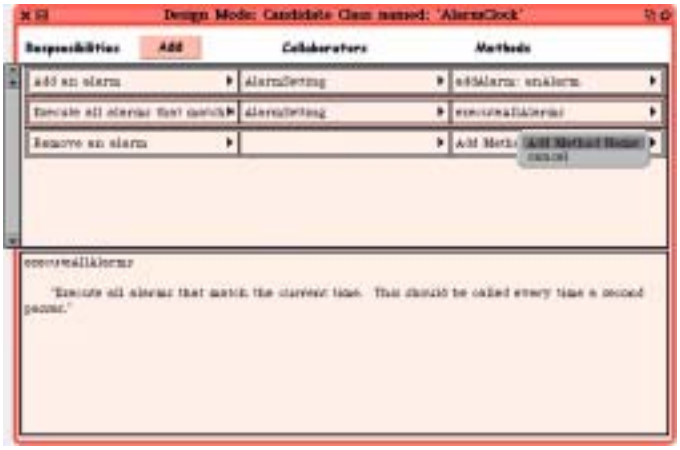


Figure 3. CRC Card Design, adding Methods.

- Program Mode –When the designer moves into Program mode, ECoDE checks all CRC Cards to insure that all Responsibilities have corresponding Methods. If any Responsibilities have no corresponding Method, the designer is notified and given the opportunity to fix the deficiency. In Program mode, ECoDE provides the designer with the ability to convert CRC Cards to actual Classes. Upon conversion to a Class, ECoDE prompts the designer to declare attributes for the Class. ECoDE then automatically generates source code "stubs" for each Class. The designer may then complete the implementation of each Method for each Class.

Figure 4. Program Mode.

Background and Related Work

CRC Cards

CRC Cards are an informal approach to object-oriented modeling. Introduced by Kent Beck and Ward Cunningham in "A Laboratory for Teaching Object-Oriented Thinking" released in OOPLSA '89, CRC Cards are represent multiple objects simultaneously.[3] Physically, CRC Cards are a set of index cards used to represent the responsibilities of classes and interaction between the classes, hence the name, Class-Responsibility-Collaboration.

Scenario-based Requirements Analysis

Critical to software design is the analysis of the requirements of the program. Software analysts must take an active role in eliciting a complete set of requirements from the user. One approach to this is to think through actual scenarios of how the system is intended to be used. One scenario-based requirements elicitation technique, the *Inquiry Cycle*[4] extends this approach by forcing the user to generate scenarios systematically, asking questions such as what might prevent a system goal from being met and what the system should do to compensate.

Goal Decomposition and Reflective Redesign

A necessary characteristic of Ectropic Design is that it must be capable of reflecting on its own behavior and reconfiguring itself. To allow for this *computational reflection*, the structure of the system must be explicitly represented. The Task-Method-Knowledge (TMK) representation [6] is organized around a functional decomposition of the system's goals. Tasks correspond to goals, methods describe the code used to accomplish a task, and a domain model describing system actors and important data types depicts knowledge. This reflection component can directly relate end-user tasks to system components.

Domain of this Study

The domain of this study was limited to testing the usability and performance of ECoDE as well as its service as a design tool. We were also interested in the value of Ectropic Software Design as an educational aid.

Given our overall objective of servicing the Open Source development community, we identified that students of software design would provide an analogous group to test our research hypotheses. Open Source developers are most interested in maximizing the expediency in getting the job done. Students have similar motivations in that they are most focused in simple getting their assignments completed. Thus, we chose students as reasonable test subjects for the larger Ectropic goals.

Our main focus was to investigate how student software designers understand the design process. In particular, we focused on how they subdivided the goals of a program, identified responsibilities, and distributed responsibilities throughout collaborating classes. ECoDE is a tool that assists the designer in these specific components of software design. By using ECoDE in our testing, our objective was to encourage the use of CRC Cards and Scenario-based analysis during the analysis and design of software. The experiment also provided a vehicle for testing the effectiveness of Ectropic Design in the software design process and our hypothesis that ECoDE provides a desirable design environment.

One of the challenges of teaching novice programmers software design is the difficulty in getting the students to focus on design concepts, rather than on implementation concerns. ECoDE, as an educational aid, applies *constructionism* educational theory, invented by Seymour Papert. Papert believed that learning occurs "more felicitously" when individuals construct something meaningful to themselves or to others around them, things that are public, i.e., can be shared, evaluated, and critiqued.[5] Individuals are more motivated to solve problems because they are constructing things that others may see and use.

ECoDE requires explicit discussion and specification of design components. In comparison to students learning more traditional software design processes, our expectation was that this would result in the student obtaining a better understanding of the design process in general. In addition, we expected an increased comprehension of the structure and purpose of the components of the student's design and the interactions and relations of these components.

Research Hypotheses

This research makes the following hypotheses:

- ECODE provides a *desirable* design environment.
- Students using ECODE will obtain a better understanding of the design process and a more clear understanding of the structure and purpose of the components of the software design, and the interactions and relations of these components, than students learning traditional software design processes.

Experiment Process

Introduction to Experiment Process

The experiment compares the outcomes of a control group and a subject group.

The control group consisted of fifty-two volunteer students enrolled in CS 2340, Objects and Design, during Fall Semester, 2001. An announcement was posted on the class CoWeb soliciting volunteers from the class. Participating students were awarded extra credit points to be applied to their final semester grades (Other extra credit options already exist in the class). Participants were given a questionnaire asking for a description of their experience with computer software design (Appendix A). The questionnaire included some basic design questions that served as a basis of their understanding of the design process. Participants next complete an exercise in which they described and evaluated the design of their semester project in progress at the time.

The subject group consisted of approximately two hundred students enrolled in CS 2340, during Spring Semester, 2002. At the beginning of the semester, these students were given a questionnaire asking for a description of their experience with computer software design (Appendix A). The questionnaire included some basic design questions that served as a "pretest" of their understanding of the design process. The following activities were a part of the course requirements:

- Participants completed an initial design of software to solve a given problem using ECoDE. This was a semester project that encompassed the software development life cycle.
- Participants completed design enhancements using ECoDE.
- Participants were required handle a surprise requirement change that required another major design change, using their choice of design processes and tools.

Participants then completed a "posttest" and a questionnaire regarding the usability of the prototype. The posttest included an exercise in which they were asked to describe their in detail.

Evaluation

The following evaluation methods were used to resolve the research questions:

Experiment Design

A subjective evaluation of the experiment design was performed to quantify the following:

- the effectiveness of the preliminary questionnaire in capturing the level of the participants' experience in and understanding of software design.
- the effectiveness of the design description and evaluation exercise in testing the students' understanding of the design process and the structure and purpose of the software,
- the effectiveness of the use of the Ectropic CoWeb for reporting of defects and discussion of usability issues, and
- the effectiveness of the built-in logging features of ECODE in representing the participants' behaviors throughout the experimental period.

Evaluation of Design Environment

In the ECODE environment, five instruments were used to evaluate the environment:

- Ectropic CoWeb for reporting of defects – students were asked to report all defects through a link to the Ectropic CoWeb from the class CoWeb site (http://coweb.cc.gatech.edu/cs2340/2102)
- Ectropic CoWeb for reporting of usability issues - students were asked to openly discuss usability questions, comments, and suggestions through a link to the Ectropic CoWeb from the class CoWeb site (http://coweb.cc.gatech.edu/cs2340/2107)
- Log files of students' interaction with ECODE – ECODE maintained a log file for each participant to capture specific activities and to represent the participant's design behaviors while using ECODE.
- Students were given a usability questionnaire at the conclusion of the experiment.
- Results of students' choice of design environments for redesign exercise.

Evaluation of Student Designs

Student designs were evaluated subjectively and comparatively as follows:

- A subjective analysis of the student designs was performed by applying accepted principles in Object-oriented analysis and design. (Appendix G)

Evaluation of Ectropic Design as an Educational Aid

A comparative analysis of the results of the posttests of the subject group and the exercise completed by the control group was performed to evaluate whether the subjects' understanding of the design process and of the structure and purpose of the software they designed was enhanced respective to the control group.

RESULTS AND FINDINGS

ECoDE Usability Issues

Usability of ECoDE was evaluated using two methods:

- Ectropic CoWeb for reporting of usability issues - students were asked to openly discuss usability questions, comments, and suggestions through a link to the Ectropic CoWeb from the class CoWeb site. (http://coweb.cc.gatech.edu/cs2340/2107)
- Students were given a usability questionnaire at the conclusion of the experiment (Appendix B).

Ectropic CoWeb

Initially, while students were first learning to use ECoDE, a few students reported usability issues on the Ectropic CoWeb. Students expressed frustration in having to learn how to use ECoDE, and that they were more comfortable with pencil and paper. An interesting discussion ensued regarding CRC Cards and how useful they were as physical index cards versus electronic cards. Several students agreed that there was an notable value in having the physical cards available while working together in groups. Cards may be passed around, grouped together in scenarios for interaction and collaboration. While students acknowledged the potential value of electronic cards in open source development with totally disconnected designers, some pointed out that this was an educational setting, they were physically working together in groups, and thus that benefit was not relevant to them.

Several specific suggestions were provided that were helpful and should be considered in future versions of ECoDE:

- Organize and size the views of opened CRC Cards and Scenarios such that they do not overlap
- Importing and exporting objects such as CRC Cards or Scenario so that they may be passed around individually to team members
- More graphical representations to influence the user experience

Final Questionnaire

Students were asked to provide comments regarding their experience with ECoDE in a final questionnaire (Appendix B). An important indicator of the usability of ECoDE and our hypothesis that ECoDE is a desirable design environment was the decision of the student whether to use ECoDE for the final two design submissions, for it they were free to choose not to use ECoDE. It is interesting to note that 68.5% of the students voluntarily chose to use ECoDE. The top reason given for choosing to use ECoDE was it's simplicity in update their previous design versions. Of the students who chose not to use ECoDE, the reason most frequently given was that they preferred pencil and paper (44%).

The following reasons were provided for those choosing to use ECoDE:

- Simple Updating 51%
- Helpful, Easy to Use, Fast 21%
- Thought it was required 10%
- Completeness of Design Analysis 8%
- No Reason Provided 10%

The following reasons were provided for not choosing to use ECoDE:

- Preferred Paper 44% (8/18)
- No Changes in Design 22% (4/18)
- Time Consuming 17% (3/18)
- Too Many Bugs 11% (2/18)
- No Reason Provided 6% (1/18)

How the Students Responded to an Automated Design Tool

Interestingly, all comments were posted at or near the time of the due date for the first design submission using ECoDE. No further comments were posted for future design submissions and very few verbal comments were communicated. This may indicate that the initial time investment required to learn how to use ECoDE frustrated the students initially. Having little to no experience in actually software design processes, coupled with the course requirement of learning a new programming language, clearly frustrated many students. These students were much more concerned with implementing a working program than following a specified design process to insure good design and good documentation. The user interface, while extremely simple in design to the researchers, was somewhat challenging to several of the students. However, after working with the tool for a while, the tool appears to have become more of an aid and less of a handicap.

Another point to consider is the timing in which the tool was introduced to the subject group. Most students had already completed preliminary design work and significant implementation prior to ECoDE's introduction and required use. Had the tool be available to them immediately, it is likely that many of the frustrations and comments regarding doubling their workload could have been avoided and a truer test of the effective of ECoDE could have been achieved.

Capturing Design Tasking Scripts

ECoDE maintained a log file for each participant to capture specific tasking activities and to represent the participant's design behaviors while using ECoDE. The following information about each action was logged:

- Date
- Time
- Author
- Sender Class Name
- Sender Object Name
- Action
- Receiver Class Name
- Receiver Object Name
- Mode

In order to analyze this data properly, the following procedures were utilized:

- Individual text files were scanned using an *awk* script and a graphical representation of behavior was generated in order to characterize similar tasking scripts within the data.

- Individual text files were imported in an Excel spreadsheet and analyzed with further scrutiny to detailed the specific tasking paths followed

Validating Data with Awk

The *awk* utility in UNIX interprets a special-purpose programming language that makes it possible to handle simple data-reformatting jobs easily with just a few lines of code. This was very helpful in validating the data and characterizing similar tasking scripts within the data. A random visual perusal of the log files indicated that some of the submissions clearly showed that the student's interaction with ECoDE occurred either after implementation actually occurred, or after for purely perfunctory reason due to the course requirement to submit designs using ECoDE. We wanted to identify these special cases and separate them from the actual detailed analysis. Using *awk*, each individual team's submission was scanned first for the *action* of creating either a CRC Card, a Scenario, or an Episode, and secondly for marking a column to distinguish which of these three objects were created.

The following *awk* script was used to create a graphical representation of the tasking path followed by the designer:

```
BEGIN { FS = "|" }
/creation/ && $5 ~ / *CRCCard */ {printf "X\n" }
/creation/ && $5 ~ / *Scenario */ {printf "\tX\n" }
/creation/ && $5 ~ / *Episode */ {printf "\t\tX\n" }
```

Figure 3 to the right is an sample visualization produced by the *awk* script. The three columns represented the creation of CRC Cards, Scenarios, and Episode, respectively. By created this textual table, the path followed by the student could easily be visualized. Interestingly, most student tasking paths matched one of the following five patterns:

- *Pattern 1 –* *Scenario → Episodes → CRC Cards*
  User created one Scenario, began building the scenario by creating Episodes (CRC Card - Responsibility pairs), creating CRC Cards as they were identified as necessary. Then moved on to create new Scenarios, again working through each one systematically. Figure 3 is an example of this pattern. Other sample visualizations are presented in Appendix C.

- *Pattern 2 –* *Scenario List → CRC Card List → Episodes → CRC Cards*
  User created several Scenarios, then created several CRC Cards, then worked through Scenarios, building them by creating Episodes, adding a new CRC Cards as necessary. Sample visualizations are presented in Appendix D.

- *Pattern 3 –* *CRC Cards → Scenario → Episode → CRC Cards*
  User created a few CRC Cards, then created a Scenario, worked through the Scenario, building it by creating Episodes, adding more CRC Cards as necessary. Then moved on to create new Scenarios, again working through each one systematically. Sample visualizations are presented in Appendix E.

- *Pattern 4 –* *CRC Card List → Scenario List → Episodes → CRC Cards*
  User created most of their CRC Cards first, then created a list of Scenarios, then worked through the Scenarios, building by creating Episodes, adding a couple new CRC Cards as necessary. Sample visualizations are presented in Appendix F.

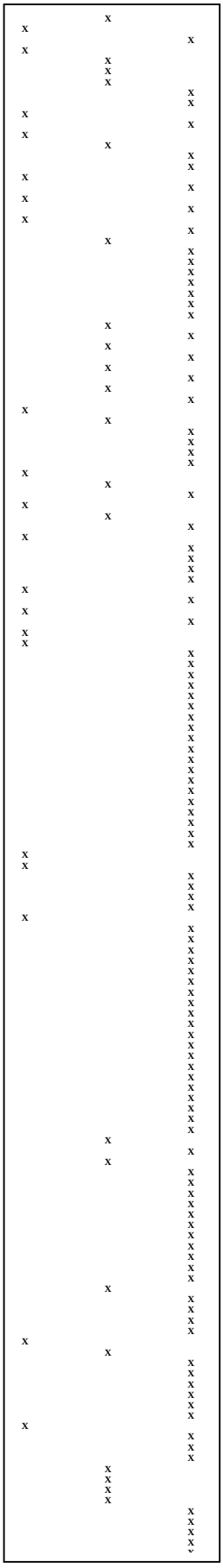


Figure 4. Pattern 1

- *Pattern 5 –* These submissions were eliminated due to various reasons. Examples include clearly bad data, creation of a CRC Card list and a Scenario list but little else, no Scenarios at all, etc.

Based on this analysis, the 36 distinct team design session submissions were categorized as follows:

Pattern 1	3 Teams
Pattern 2	6 Teams
Pattern 3	14 Teams
Pattern 4	10 Teams
Pattern 5	3 Teams

Detailing Design Behavior

Once particular patterns were established for each team submission, detailed design behavior was analyzed using Excel spreadsheets to further examine the specific paths followed. The specific paths were analyzed and compared to their actual project submissions using ECoDE. For example, the following script shows the path that a particular team followed through their design session

Subject Design Session: Pattern 3

February 20th
Created new EctropicDesign
Created 9 new CRC Cards
Created 6 new Scenarios
Worked through several CRC Cards creating and adding basic responsibilities
Moved to Scenarios
Created Episodes (Responsibility/CRCCard pairs)
Created 2 new CRC Cards as Collaborators
Reassigned Responsibilities
Completed Scenarios
Rejected 3 CRC Cards
Renamed 2 CRC Cards
Moved to Design Mode
More work on CRC Cards – creating and adding methods
Adding new responsibilities
Removing methods
Created 1 new CRC Card
Reassigned several responsibilities
Saved Design using menu item
Saved Project using menu item

February 21st
Start of new session
More work on CRC Cards – creating and adding methods
Removed several responsibilities
Reopened Scenarios
Created 7 new Episodes
Switched back to Analysis Mode
Created 2 new CRC Cards
Reassigned several responsibilities, adding collaborators
More work on new CRC Cards – creating and adding responsibilities
Switch to Design Mode
More work on new CRC Cards - creating and adding methods
Saved Design using menu item
Saved Project using menu item

End of Design Work for Milestone 3 (due February 26th)
This team's P3 turnin indicates implementation occurring prior to design and from 2/21 – 2/26. This is consistent with expectations.

The path this team followed clearly showed Scenario-based reasoning in their analysis of the responsibilities necessary to meet the program goals as represented in specific Scenarios. It also showed the ability of ECoDE to provide a design environment that allows flexible movement between CRC Cards and Scenarios throughout the design process. This team easily reassigned responsibilities and comfortably moved from one mode to the next.

Student Design Submissions

Students from both the control group and the subject group were asked to describe their final designs with as much detail as possible. These design submissions were each scored using a specifically designed scoring system based on generally accepted object-oriented design principles (Appendix G). The evaluation was divided into three separate parts:

- Design Evaluation – this evaluation focused on the actual design as an example of good object-oriented software design. The following criteria was scored:
  - Object-Oriented CRC Cards/Classes (Names
  - Division of Responsibilities, Control, and Communications
  - Reasonable Level of Information Access
  - Reusability Demonstrated
  - Good Use of Inheritance
  - Correct/Sensible Relationships
  - Correct/Sensible Concepts
  - Sufficient Design Detail

- Demonstrated Understanding of the Design Process – this portion of the evaluation focused on certain markers that indicated the subjects understanding of the design process. The particular criteria evaluated included:
  - Design before implementation
  - Requirements analysis
  - Good Use of CRC Cards
  - Good Use of Scenarios

- Evaluation of Design Documentation – this portion of the evaluation focused on the specific design documentation submitted. Students were asked to provide detailed documentation but were not given specific requirements as to what was sufficient. They were permitted to submit whatever documentation they chose to submit. The following criteria was scored:
  - Complete Class Descriptions
  - Use of UML Class Diagrams
  - Use of Other UML Diagrams
  - Clear Detailed Documentation of Attributes
  - Clear Detailed Documentation of Services
  - Clear Detailed Documentation of Associations
  - Use of Scenarios/Event Traces
  - General Completeness of Documentation Submitted

Only design documentation specifically submitted for this evaluation was evaluated and the designs were evaluated independent of source code or even ECoDE files. For all criteria, if nothing could be ascertained from the subject's submission, the criterion was scored as a 0. For all criteria, only exception use or demonstration of the criterion was given a score of 5.

Subject Group Use of Scenarios

With regard to a quantitative comparison of the scoring between the control group and the subject group, there was only a marginal increase in average scores in the design evaluation. No information was obtained from the control group regarding their understanding of the design process. Thus no comparison could be made. However, subjectively, the subject group documentation was much more detailed in nature. Many of the subject group students demonstrated a clearer personal understanding of their actual design. The control group was asked specifically to describe their design regard to a given usage scenario. Scenarios were not specifically address with

the subject group. However, the subject group more often chose to use scenarios as a way to describe their software design and clearly demonstrated good usage of scenarios as analysis and design tools.

Analysis of Experimental Process

In hindsight, some difficulty was experienced in the final analysis of the data collected. The log files were extensive, and much work could have been avoided given a better approach to collection and a better design of the specific log entries. Specifically, the following modifications are recommended for further testing:

- An EctropicDesign object should maintain separate log files for specifically targeted events such as creation of objects, modal task tracking, tracking the assignment and reassignment of responsibilities, design critique events, and error handling.

- Log entry date and time data should be separated such that more detailed analysis could be performed regarding the time spent by the test subject on specific design tasks such as, scenario-based analysis, assignment of responsibilities and corresponding methods, and code generation.

- Instrumentation should be built to perform matching on actual implementation and design components, to assist in the determination of the order in which components were designed versus implemented.

The design exercise both the control group and the subject group were asked to complete at the end of the semester appears to have been too loosely structured. They were explicitly designed to provide very little instruction on what would be considered acceptable documentation. As a result, many of the submissions were insufficient in detail. While the conclusion could be made that is an indication that the student demonstrated a lack of understanding of the necessity for complete design documentation, it is more likely the case that they were not provided with sufficient instruction. It should have been explicitly stated that the submitted design documentation should be fully implementable solely from the documentation and thus must be complete and presented in generally acceptable formats. Additionally, each group should have been explicitly asked to describe in detail their individual design processes, what specific tasks they personally contributed to, and what difficulties they experienced.

Additionally, no data was collected from the control group to allow them to demonstrate their use and understanding of software design processes and thus no comparison could be made to the subject group work.

The greatest question posed by the experiment design is the challenge in adapting an automated design tool to a course designed specifically to teach software design. Having little to no experience in actually software design processes, coupled with the course requirement of learning a new programming language, clearly frustrated many students. These students were much more concerned with implementing a working program than following a specified design process to insure good design and good documentation. The user interface, while extremely simple in design to the researchers, was somewhat challenging to several of the students. Another point to consider is the timing in which the tool was introduced to the subject group. Most students had already completed preliminary design work and significant implementation prior to ECoDE's introduction and required use. Had the tool be available to them immediately, it is likely that many of the frustrations and comments regarding doubling their workload could have been avoided and a truer test of the effective of ECoDE could have been achieved.

Conclusions and Future Work

The results of this experiment do support the original research claims that:

- ECODE provides a *desirable* design environment.
- Students using ECODE will obtain a better understanding of the design process and a more clear understanding of the structure and purpose of the components of the software design, and the interactions and relations of these components, than students learning traditional software design processes.

However, while 68.5% of the students voluntarily chose to use ECoDE for the final design submissions for which it was not required, the major reason given was the simplicity ECoDE offered in updating their original designs. The initial frustrations the students experienced in learning how to use ECoDE, coupled with the timing of the introduction of ECoDE appears to have had a great impact on the acceptance of the tool by the students. Several student volunteered comments at the end of the semester. It is interesting to note that the negative comments focused on the usability of ECoDE claiming it was inflexible, created more work, and lacking sufficient collaborative support. However, the positive comments focused on how useful ECoDE's Scenario Navigator was, that is was a great brainstorming tool and forced them to conduct very detailed design analysis. Clearly the negative comments were likely from those student who experienced a high level of frustration with the interface initially.

As to the effectiveness of ECoDE in teaching object-oriented design and design processes, clearly the subject group demonstrated a better understanding of the structure and purpose of the components of their final designs. They demonstrated far greater use of scenario-based reasoning in their requirements analysis and the final design submissions tended to show a better understanding of the interactions and relationship between the objects incorporated in their designs.

A significant benefit of usage of ECoDE as a design tool that surfaced during the course of this experiment is it's support for easy design documentation maintenance. Further research should be completed to properly analyze this benefit. An example experiment would be to test ECoDE with two separate groups: A control group completing design work without using ECoDE, maintaining hard copy design documentation, an indicated preference of many of this experiment's test subjects and a subject group completing design work with ECoDE. An assignment requiring multiple design modifications would sufficiently test this proposed benefit.

Future research is necessary to fully test ECoDE's effectiveness as an educational aid. The lack of collaborative support was clearly apparent in the finding of this experiment. Many students indicated that ECoDE would provide good design support coupled with sufficient collaborative support. This is clearly the next phase of development for Ectropic Design. Once completed, an experiment to test ECoDE effectiveness as a *collaborative* design tool should be conducted. Principal research questions to be answered would include the ability of ECoDE to maintain conceptual integrity through a collaborative design process, and the users' ability to adapt to the Ectropic environment.

References

[1] Rugaber, Spencer and Guzdial, Mark. *Ectropic Software*. International Conference on Software Engineering (ICSE-99) Workshop on Software Change and Evolution. Los Angeles, May 17, 1999

[2] Eric Raymond. "Introduction to Open Source." http://www.opensource.org/intro.html

[3] Cunningham, W. and Beck, K.: " A Laboratory for Teaching Object-Oriented Thinking," in Proceedings of OOPSLA-89, October 1989.

[4] Colin Potts, Kenji Takahashi, and Annie I. Anton. "Inquiry-Based Requirements Analysis." *IEEE Software*, 2(11):21-32, March 1994.

[5] Harel, Idit and Seymour Papert (Eds.). *Constructionism*, Norwood, NJ: Ablex, 1991.

[6] J. W. Murdock and Ashok Goel. "A Functional Modeling Architecture for Reflective Agents." *AAAI-98 Workshop on Functional Modeling and Teleological Reasoning*, Madison, WI, July 1998.

Appendix A. Preliminary Questionnaire

Ectropic Software Design Study – Preliminary Questionnaire SUBJECT #_____

Part 1: Personal Information

Class Rank:

- ☐ Freshman
- ☐ Sophomore
- ☐ Junior
- ☐ Senior
- ☐ Other

Overall GPA:

- ☐ < 2.0
- ☐ 2.0 – 2.5
- ☐ 2.5 – 3.0
- ☐ 3.0 – 3.5
- ☐ > 3.5

What letter grade do you expect to receive for CS 2340? _____

Major _____

Part 2: Background

Is this your first educational course in Computer Software Design? _____

If not, please list other educational courses you have taken related to Computer Software Design:

_____

_____

_____

How many years have you been programming? _____

Please describe any software development experience you have exclusive of a classroom environment:

_____

_____

_____

Appendix B. Final Survey

Ectropic Final Questionnaire (Spring 2002) **gt #_____**

1. **For milestones P3 through P5, you were required to use ECoDE to design your Campus Map System. Considering your preliminary design work prior to ECoDE's release, which of the following best describes your group's use of ECoDE for each milestone:**

check at least one for each milestone

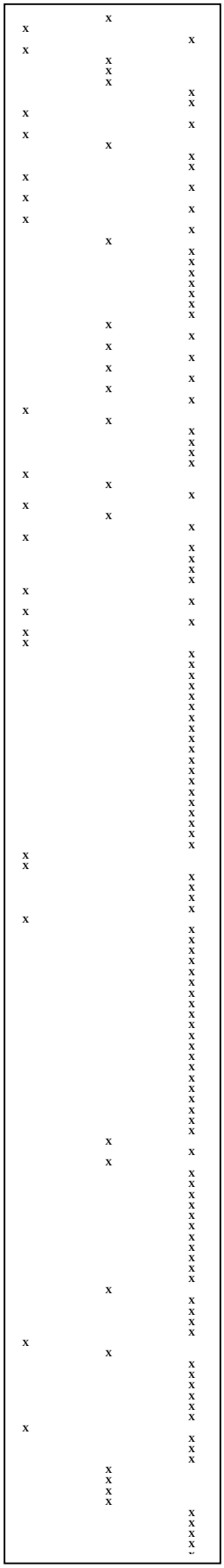
		P3	P4	P5	P6	P7
1	We used ECoDE for all design sessions to help us build and optimize our designs Comments:					
2	We used ECoDE only after working out design solutions together on paper Comments:					
3	We used ECoDE to help us document our design modifications Comments:					
4	Scenarios in ECoDE helped us determine Classes and Methods required How?					
5	ECoDE inhibited the design process for us. How?					
6	Did not use ECoDE for this milestone					

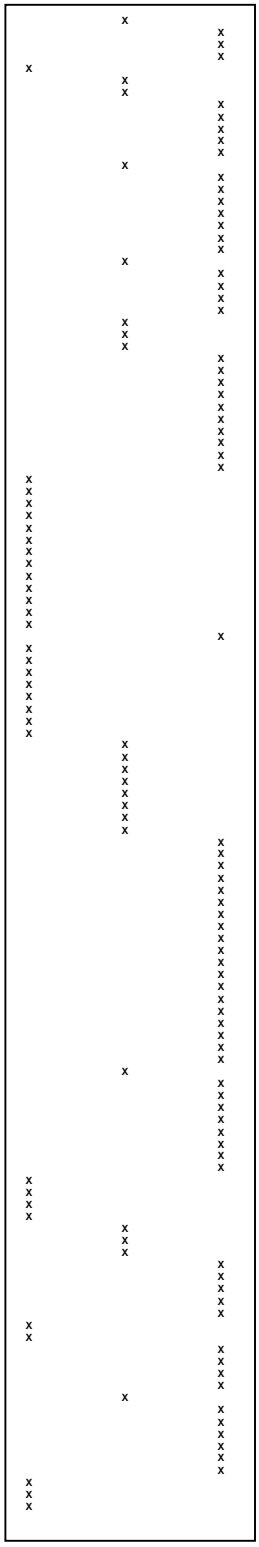
2. **Did you continue to use ECoDE for milestones P6 and P7? Why or why not?**

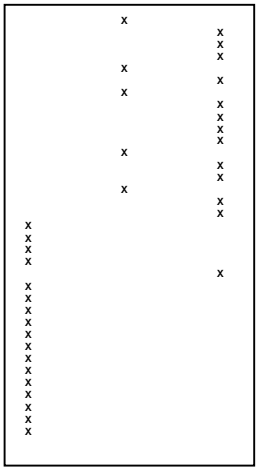
3. **Did you utilize the ECoDE User's Manual and/or Alarm Clock Example provided? If so, did you find it helpful? Why or why not?**

4. **Please provide any additional comments, feedback, suggestions, etc. Please be candid and as specific as possible. Thank you!**

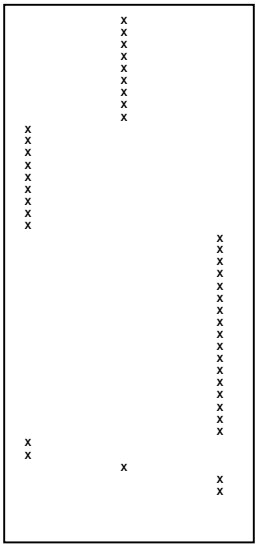
Appendix C. Pattern 1 Visualizations

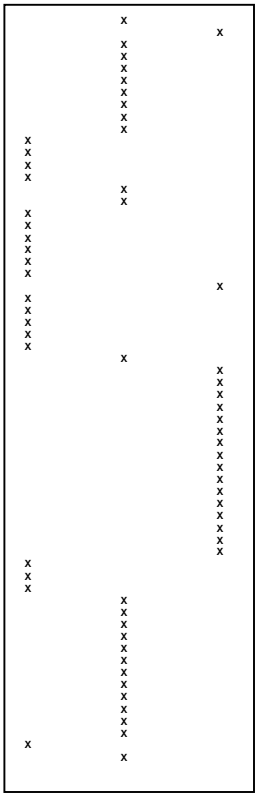


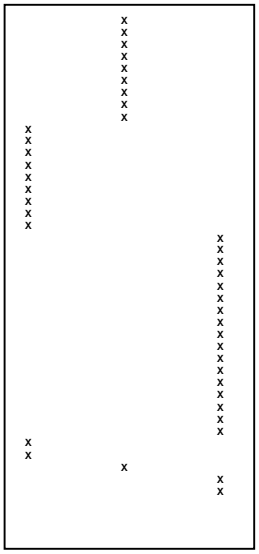




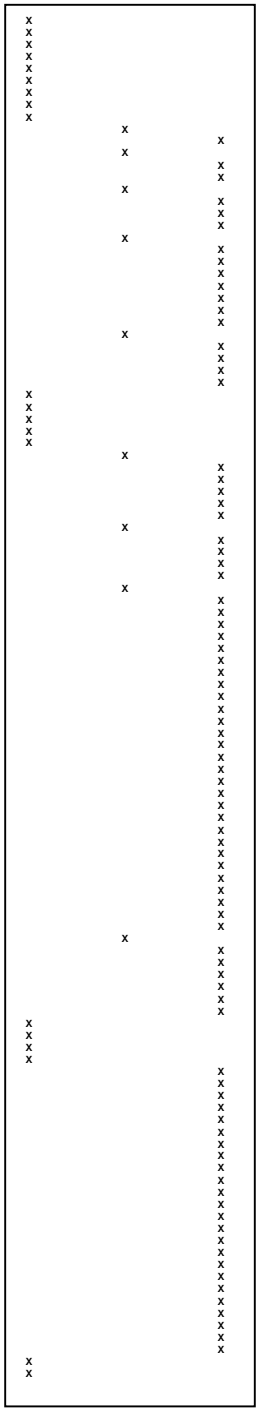
Appendix D. Pattern 2 Visualizations

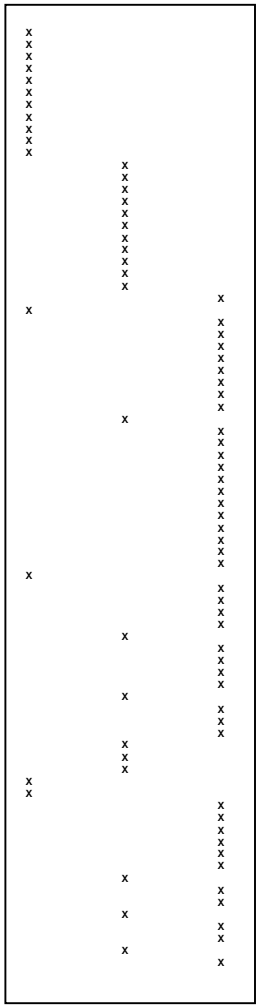


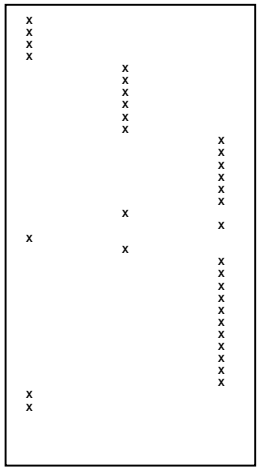




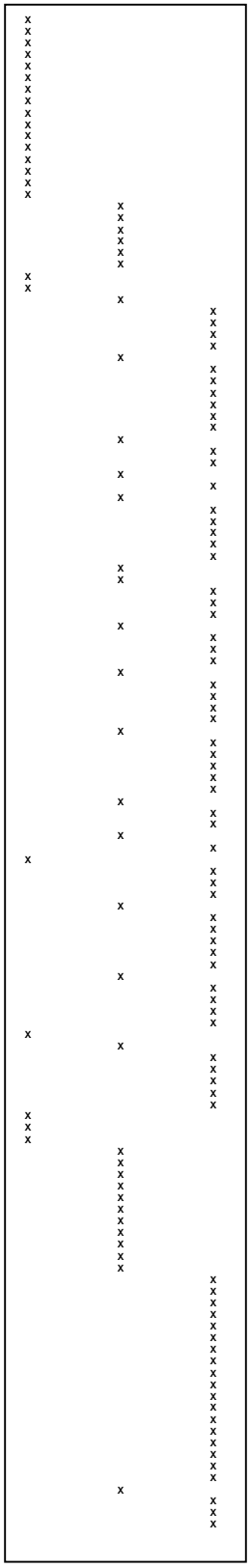
Appendix E. Pattern 3 Visualizations

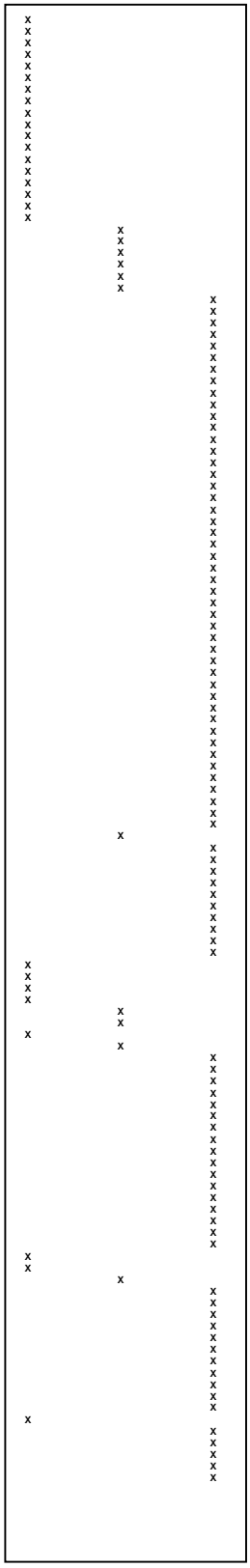


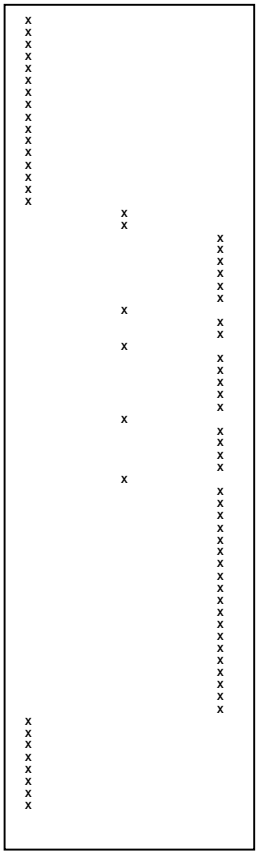




Appendix F. Pattern 4 Visualizations







Appendix G. Student Design Coding

Ectropic Object-Oriented Software Design Evaluation

(Designed specifically for the CS2340 student)

Subject # _____ **Team Name** _____

Design Evaluation

	Criteria	Scoring	Score
1	**Object-Oriented CRC Cards/Classes (Names)** *No Info or None (0) . . . Complete (5) Based on percentage of CRC Cards/Classes*	5	
2	**Division of Responsibilities, Control, Communications** *No Info or None (0) . . . Excellent Sensible Distribution (5)*	5	
3	**Reasonable Level of Information Access** *No Info or No Attention Apparent (0) . . . (5) Too Much/Little . . . Reasonableness*	5	
4	**Reusability Demonstrated** *No Info or No Attention Apparent (0) . . . (5) Based on percentage of opportunities realized*	5	
5	**Good Use of Inheritance** *No Info or No Attention Apparent (0) . . . (5) Based on percentage of opportunities realized*	5	
6	**Correct/Sensible Relationships** *No Info or No Attention Apparent (0) . . . All relationships make sense- no missed opportunity (5)*	5	
7	**Correct/Sensible Concepts** *No Info or No Attention Apparent (0) . . . (5) General subjective score on design concepts*	5	
8	**Sufficient Design Detail** *No Info or No Attention Apparent (0) . . . (5) General subjective score on design detail*	5	
	Design Evaluation Subtotal	**40**	

Demonstrated Understanding of the Design Process

	Criteria	Scoring	Score
1	**Design Before Implementation Demonstrated** *No Info or No Attention Apparent (clearly afterward) (0) . . . Extensive Preliminary Detail (5)*	5	
2	**Requirements Analysis/Tracing** *No Info or No Attention Apparent (0) . . . Extensive analysis (5)*	5	
3	**Good Use of CRC Cards** *No Info or No Attention Apparent (0) . . . Rejected Cards, Renamed Cards . . . (5)*	5	
4	**Good Use of Scenarios** *No Info or No Attention Apparent (0) . . . Extensive Scenario Investigation (5)*	5	
	Design Process Subtotal	**20**	

Evaluation of Design Documentation

	Criteria	Scoring	Score
1	**Complete Class Descriptions** *No Info or No Attention Apparent (0) . . . All Attributes, Services, Details (5)*	5	
2	**Use of UML Class Diagrams** *No Info or No Attention Apparent (0) | Made attempt (1) . . . Complete and Correct Usage (5)*	5	
3	**Use of Other UML Diagrams** *No Info or No Attention Apparent (0) . . . (5) One point for each additional UML diagram up to 5*	5	
4	**Clear Detailed Documentation of Attributes** *No Info or No Attention Apparent (0) | Attributes listed (1) . . . Complete w/detailed description (5)*	5	
5	**Clear Detailed Documentation of Services** *No Info or No Attention Apparent (0) | Services listed (1) . . . Complete w/detailed description (5)*	5	
6	**Clear Detailed Documentation of Associations** *No Info or No Attention Apparent (0) | Some indication (1) . . . Clear depiction (5)*	5	
7	**Use of Scenarios/Event Traces** *No Info or No Attention Apparent (0) | Scenarios provided (1) | Multiple usage (2) . . . Super use (5)*	5	
8	**General Completeness of Documentation Submitted** *No Info or No Attention Apparent (0) . . . (5) General subjective score of completeness*	5	
	Design Documentation Subtotal	**40**	
	Final Scoring TOTAL	**100**	