A Case Study of Ectropic Design

Isabel Twyeffort

December 1, 2004

1 Introduction

As a program is developed over time, it is tempting for the developers to add every neat new feature that is requested. However, some features may move the program away from its original purpose. Through the use of design tools, it may be possible to determine how greatly new features diverge from the original focus of the program.

ECoDE is such a design tool, using goals, scenarios, and CRC cards to assist a system's designers with creating a new system. However, it can also be used to modify an existing design, for instance, to add new features. We are interested in how evident divergent features will be in this system.

2 Plan

We intended to test the capabilities of Ectropic Design with respect to the addition of new features by reverse engineering a sample program from Sourceforge. The subject was to be described in the design and analysis portions of ECoDE and then modified. The extent to which goals, CRC cards, and responsibilities changed was to be an indicator of the divergence of the proposed modification.

3 Results

The subject I selected was named ListfulThinking. This was a simple Java program designed to assist with managing todo lists. Its special feature is the system it uses for determining the priority of a given task; it infers this from other information provided by the user, rather than a directly supplied priority value. Apart from this, its design and functionality are relatively straightforward.

Although I originally selected a Java program specifically so that I could read the code, this proved unnecessary; I was able to deduce the functionality sufficiently to be described in Ectropic Design using goals and scenarios. From these the CRC cards and responsibilities could be derived. Fortunately, the design of this program was simple, as I had intended for a first attempt at using Ectropic Design on an existing program. (My suggestion for a second subject to analyze was declined, with the explanation that greater depth might be more useful than greater breadth.)

Because of the simplicity of the program, the high level goals were clear, as was one possible set of objects (corresponding to CRC cards). The one highest level goal was determined to be 'Organize todo tasks'. This goal led to subgoals such as 'Create new task' and 'Add task to list'. The CRC cards were 'List', 'ListItem', 'ListManager', and 'ListGUI'. The scenario list was much larger, including approximately ten scenarios.

All features requested on the Sourceforge request list were reasonably in line with the existing features and goals of the program; none of these requests caused any observable changes in either goals or CRC cards. One of these requests was the ability to move a given task from one list to another. This request could be modeled in Ectropic Design entirely with pre-existing responsibilities, indicating that it would likely require only a small amount of work in the actual application to implement. This also demonstrated that the request included no new concepts in the design.

Adding the ability to have the program alert the user at preset times caused much more change in the high level design of the program. In addition to requiring new responsibilities, which are roughly equivalent to methods, this feature required adding a new goal tree and new CRC cards. The new goal tree was rooted at 'Alert user to time' and the new CRC cards included 'TaskAlert'.

4 Discussion

Given an existing design, it does seem as though the goals, especially, are sensitive to shifts in a program's purpose. However, it is difficult to tie these to the rest of the design. The CRC cards also indicated major changes in a program's purpose, and were easier to work with as they were more closely tied to the scenarios. For instance, both the goals and the CRC cards were modified in the intentionally inappropriate feature addition; however, the goals were more sensitive to lesser modifications such as adding deletion of lists.

Perhaps one of the more important uses of this tool is that with the integration of goals into the design, it forces the designer to actually think about what the program is designed to do as opposed to simply what it could be made to do. Because they are forced to actually think about how a feature fits into the overall design of a program, designers are less likely to allow feature creep to overcome their projects.

5 Conclusions and future work

A closer connection between the goals and scenarios in ECoDE's design phase could make it easier to detect changes in a design's purpose. This connection could possibly be implemented by creating a new critic that ensures that all responsibilities are assigned to goals. Another possibly useful critic could ensure that all leaf subgoals have responsibilities. Finally, it might be possible to connect scenarios or responsibilities to goals directly, similarly to how responsibilities are connected to CRC cards.

The existing critics in ECoDE provided little information about the coherence of the design. Thus, this work was entirely manually performed. Automation of such critique would be essential to performing this analysis on a larger

program. However, analyzing larger programs would be helpful in determining the practicality of this method for normal programs, given that ListfulThinking was abnormally simple.