Visualizing Non-Functional Traces in Student Projects in Information Systems and Service Design

Jessica Bolger Voytek
School of Information
University of California, Berkeley
102 South Hall, Berkeley, CA, 94720
jvoytek@ischool.berkeley.edu

Julián Limón Núñez
School of Information
University of California, Berkeley
102 South Hall, Berkeley, CA, 94720
limon@ischool.berkeley.edu

ABSTRACT
In this paper we describe a visualization technique that was developed to help students and instructors keep track of the relationships between important observations and key insights in design activities.

Author Keywords
Traceability, visualization, design, trees, graphs.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Design, human factors.

INTRODUCTION
At the UC Berkeley School of Information there is a graduate level course called Information Systems and Service Design (ISSD). One key concept students must master as part of this course is “traceability”, however, the concept of traceability in this context is different from previously studied traceability in the software development process. Traceability in ISSD is a software design tool, rather than a software development tool. It is the way in which a student group identifies and keeps track of the relationships between important observations and key insights from early to later design activities. It is a tool for making more objective decisions about system and service design.

Some visualizations for requirements have been developed both in business settings (http://www.ibm.com/developerworks/rational/library/07/0605_hovater/) and in academia (http://www.springerlink.com/index/3076m5105125773p.pdf). However, they are based on cryptic requirement codes, do not reflect the changing nature of requirements, and are not always associated with the specific source of the requirement (a quote in an interview transcription or an observation, or a technical constraint specified in a Service Level Agreement), or the series of documents that went into their making. These cryptic codes lose touch with the reasons for which the requirement was generated.

PROBLEM
Pinheiro defines traceability in two different categories, “Functional” and “Non-functional” traces. Functional traces are those that requirements engineers might be more familiar with: traces that are unambiguously tied to the construction of the original artifact (Pinheiro, 2003). These traces may be generated from transcribing the audio recording of an interview with a client, or identifying the system requirements of a particular piece of software that must integrate with the product you’re building. Functional traces are particularly useful for change impact analysis and requirements generation. Non-functional traces on the other hand refer to traces of a more informal nature, which flow from a more creative process (Pinheiro, 2003). These traces might arise from the interpretation of stakeholder needs from rapid ethnography, or the narrowing down of key insights from competitive analysis (Winkler, 2010). Generating non-functional traces as part of the system or service design process allows for more structured and principled decision-making in the design process.

In the student project context, non-functional traces take on an even more important role. They allow students to learn how design decisions should flow from principled investigation of the domain in which the system is being built. They also allow instructors to track student learning of these concepts.

Functional traces are well studied and visualized; non-functional traces are not. Figure 1 shows the most common traceability visualizations. These visualizations are most appropriate for describing functional traces. Figure 1(a) shows the typical pattern of traceability matrices where requirements are arranged on the x and y axes with a mark at the intersection of related requirements. Figure 1(b) shows a cross-reference type traceability diagram in which the traces are arranged in rows with a reference to related traces in the far right column (this is the way in which traceability was tracked in ISSD before this attempt).
Figure 1(c) shows a graph-based visualization of the links between requirements.

<table>
<thead>
<tr>
<th>Req 1</th>
<th>Req 2</th>
<th>Req 3</th>
<th>Class 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Req 1</th>
<th>The system shall...</th>
<th>&gt; Req 2</th>
<th>&gt; Req 3</th>
<th>&gt; Class 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req 2</td>
<td>The system shall...</td>
<td>&gt; Req 3</td>
<td>&lt;Req 1</td>
<td>&lt;Req 3</td>
</tr>
<tr>
<td>Req 3</td>
<td>The system shall...</td>
<td>&gt; Req 2</td>
<td>&lt;Req 1</td>
<td>&lt;Req 2</td>
</tr>
</tbody>
</table>

(b)

(c)

Figure 1. Illustration of the most common traceability link visualizations. Winkler, 2010 pp 542.

Each of these more common formats for visualizing traceability leave something to be desired when applied to the problem of student software design projects. The traceability matrix pattern (Figure 1(a)) can indicate a many to many relationship between requirements, but there is no inherent hierarchy or flow from one trace to the next. It can also become unreadable when more than a few requirements are shown. The cross-reference pattern (Figure 1(b)) can be modified for use in non-functional traces, allowing for more nuanced description of the relationship between traces, however the overall structure of traces is lost. One must read and understand the entire table before a picture of the overall structure of traces emerges. It would be difficult to identify traces that had been forgotten in this table structure. The graph-based visualization (Figure 1(c)) does a good job showing the over-all structure of the relationships between traces, but the relationship of traces to their parent design activity is still missing.

**DESIGN GOALS**

Analysis of the existing literature, consultation with instructors and former students, and intense debate over the possible uses of traceability in the student software design process lead us to these design goals (in order or priority):

**Encourage more systematic design decisions**

Rather than relying on intuition, providing a structured, well-designed visualization of the traceability in a student project should help students identify, and keep track of, important observations and insights that should inform later design decisions.

**Indicating the adequacy of artifacts (including prototypes, etc.)**

By tracing requirements from research activities such as competitive and stakeholder analysis, the visualization should allow students and instructors to judge the adequacy of final products (prototypes, or design plans) in addressing the needs of stakeholders (stakeholders can include individuals, groups, government organizations, and even non-human actors (i.e. existing computer systems in which this new system needs to work)).

**Facilitating audits of the design process by the teaching team**

By providing a central, organized, and consistent repository of traces, and by providing a clear overview of the design process thus far, the visualization should allow instructors to better gauge the progress of student projects.

**Informing the prioritization of requirements, “scoping”**

In student projects in particular, where the project must be completed in a short time-frame, scoping the project to its most important component parts becomes important. By allowing students to identify the most important traces, and the relationships between traces, students should be better able to scope the project to a manageable size.

**Facilitating change impact estimation and improving changeability**

When changes are necessary due to scoping or the influence of real-world constraints, the visualization should help students identify what impact certain changes would have on their project. By studying the relationship between traces in the visualization, students can better identify the impact on stakeholders of terminating or modifying trace lineages.

**THE VISUALIZATION**

The visualization we developed is a modified graph-based representation of traces throughout the student project design process. One ring on the graph indicates one design
artifact. The rings towards the center of the graph indicate artifacts that were completed before the artifact-rings towards the edges of the graph. Each node on the graph represents a trace, the lines between the nodes represent a parent-child relationship between nodes. Nodes can have multiple parents and multiple children. We also used interaction to allow viewers to better explore the relationship between traces.

In our visualization we used color in pre- and post-attentive ways. All nodes will be shaded either white, green, or red. There will be more white and green nodes than red ones, however the red nodes are of the greatest import. By using pre-attentive processing to identify the few red nodes in the graph we can call attention to “problem areas” quickly.

The particular colors we chose for the visualization came from the recommendations of Cynthia Brewer and Mark Harrower’s excellent web-based tool, ColorBrewer2 (http://colorbrewer2.org). The colors in Color Brewer 2 are based on Brewer’s previous research into the effective use of hue and contrast in visual representations. We chose a color set that is color-blind safe and “qualitative”, meaning the colors represent more nominal variables instead of sequential or diverging variables.

**Interaction**

By hovering their mouse over a node, the viewer can see the path between that node’s parents and children highlighted in yellow. By clicking on the node, the node is selected, meaning that detailed information about that node is displayed in the information section in the upper right quadrant of the graph, the path between the node and the node’s parents and children is highlighted in black, and these visual variables are fixed in the diagram until another node is selected. A visitor can then hover over other nodes to see that new node’s relationships highlighted in yellow. In this way a viewer can compare the relationships of two nodes at a time.

In addition to the overview diagram and the detailed information sections we also included a linked tag cloud in the lower right. This tag cloud shows the relative frequency of tags used in a collection of traces, and allows viewers to explore groups of traces by tag. By clicking on a tag the viewer can see traces (nodes) tagged with that term highlighted in blue.

**Color**

In addition to interaction, the colors of each node indicate nominal variables. Traces have been defined for the purposes of this visualization as either a beginning, continuation, fulfillment, or termination trace. Each type of trace is indicated using a specific color.

Color in visualization is an effective, pre-attentive way to draw attention to areas of interest in a diagram. Healy explains that pre-attentive processing is the way in which the lower level human visual system identifies characteristics of a visual scene before attention is paid to that scene. By designing systems where pre-attentive processing is used for identifying important features, a designer can make certain aspects of a visualization “jump out” at the viewer on first inspection. Hue is an example of a pre-attentive visual feature. He gives an example: a viewer searching for a red circle in a sea of blue circles will be much better able to identify the red circle than a viewer searching for a blue square in a sea of blue circles.

![Image](http://example.com/image.png)

**Figure 2. Pixel-perfect mockup of the visualization.**

A "Beginning" trace is a trace with no parents. Beginning traces will be most common at the beginning of the design process, but can be created throughout the design process and may occur more often in proximity to "termination" traces where new insights may override older insights. Beginning traces are indicated in white.

"Continuation" is the default state of each trace. If the trace has children and parents, a particular trace will be assumed to be the continuation of a lineage of traces. "Continuation" traces with no children will be highlighted in order to identify areas where ideas have been abandoned without sufficient intent (see Forgotten). Continuation traces are also indicated in white.

A "Fulfillment" trace records the completion of the goals embodied in a lineage of traces in a final artifact. The word “fulfillment” implies successful delivery, and the completion of tasks. The successful end of a lineage of traces will either be a "fulfillment" trace or a "termination" trace. Fulfillment traces are indicated in bright green.

A "Termination" trace records the intentional ending of the goals embodied in a lineage of traces. The reason for terminating a trace should be recorded in the description for that trace and could include the revision of assumptions, the addition of new data that disproves conclusions from other
traces, or the realization of real-world constraints that would limit the successful fulfillment of the goals of a trace. The successful end of a lineage of traces will either be a “fulfillment” trace or a "termination” trace. Termination traces are indicated in pale green.

Forgotten
In order to more easily identify traces that have been misplaced, students must indicate either fulfillment or termination of a particular trace lineage. If they do neither of these things the end node of a trail of traces is indicated in red, highlighting the fact that this is a trace that may have been forgotten.

DEVELOPMENT
To develop this interactive visualization we used a combination of HTML and CSS to layout the page, and Javascript (including the interactive visualization library Protovis). Because this visualization was designed to work within the context of a course website, we chose these ubiquitous web technologies.

Protovis provides methods for efficient and tidy layout of trees using the Reingold-Tilford algorithm. Reingold and Tilford’s algorithm, developed in 1981, provides an efficient and space-conscious method for laying out trees. By placing child nodes before their parents, and moving up and down through the tree sequentially, the algorithm effectively provides more space for large sibling groups.

One common problem with naïve recursive tree layouts is their inefficient use of space for depicting breadth. The Reingold-Tilford’s layout algorithm, on the other hand, is designed to maximize density (by generating tidier trees) while conserving symmetry and aesthetic pleasantness (Reingold & Tilford, 1981). The algorithm is based on four aesthetic constraints:

- Nodes at the same level of the tree lie along a straight line, and the straight lines defining the levels are parallel.
- A left son is positioned to the left of its father and a right son to the right.
- A father is centered over its sons.
- A tree and its mirror image produce drawings that are reflections of one another; moreover, a subtree is drawn the same way regardless of where it occurs in the tree.

The fourth aesthetic constraint may result in trees that are not the narrowest. However, it has been shown that it works better in terms of human perception.

Protovis’ implementation of the Reingold-Tilford algorithm is based on the modifications suggested by Buchheim, Jünger, & Leipert (2002). Buchheim et al. modifications improve the algorithm to run in linear time. This is critical for interactive visualizations, because the tree layout must be computed in real time during the interaction.

Protovis also provides a radial version of the Reingold-Tilford algorithm where radius encodes depth and arc length encodes breadth. We used this particular implementation because it makes efficient use of space. We expect students to have more traces in the latter stages of the design process (outer rings) than in the first activities (inner rings).

EFFECTIVENESS
This visualization is largely unproven at this point. We were able to get initial feedback from current and former students in ISSD, and the professor, Rober Glushko, Ph.D. and teaching team. Initial feedback has been very positive. Professor Glushko is particularly enthusiastic as he believes this will provide a more structured method for tracking design decisions in future years’ classes.

CONCLUSIONS AND FUTURE WORK
Future work in this area should expand the scope of this visualization to address the needs of larger and longer projects. Panning and zooming would be effective additions for larger projects. Also, automated generation of traces from artifacts could be a rich area for exploration.

This method for visualizing non-functional traces in student design projects is the only visualization in this space specifically designed for student design projects. The authors believe that through effective use of interaction design patterns, designing visual variables for pre-attentive processing, and effective tree layout, the visualization fulfills the needs of this particular group of users.

ACKNOWLEDGMENTS
We thank the professor, previous students and teaching assistants of the ISSD course, who provided helpful comments on our work.

REFERENCES