UNIVERSITY OF CALIFORNIA,
IRVINE

Calico: An early-phase software design tool

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Nicolas Francisco Mangano

Dissertation Committee:
Professor André van der Hoek, Chair
Professor David Redmiles
Professor Gary Olson
Professor Marian Petre

2013
DEDICATION

To my parents and sisters.
TABLE OF CONTENTS

LIST OF FIGURES vii
LIST OF TABLES x
ACKNOWLEDGMENTS xi
CURRICULUM VITAE xii
ABSTRACT OF THE DISSERTATION xiii

1 Introduction 1
  1.1 Thesis structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2 Motivation 12
  2.1 Design behaviors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  2.2 Kinds of sketches software designers produce . . . . . . . . . . . . . . . . . . 14
  2.3 How they use the sketches to navigate through a design problem . . . . . . . 19
  2.4 How they collaborate on them . . . . . . . . . . . . . . . . . . . . . . . . . . 24
  2.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

3 Research Question 28

4 Calico Version One 32
  4.1 Features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
    4.1.1 Basic features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
    4.1.2 Grid . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
    4.1.3 Scraps . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
    4.1.4 Palette . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
    4.1.5 Implementation Notes . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
  4.2 Experimental Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
    4.2.1 Recruitment and participants . . . . . . . . . . . . . . . . . . . . . . . 48
    4.2.2 Procedure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
    4.2.3 Task . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
    4.2.4 Measures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
  4.3 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
    4.3.1 Feature use . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
6.4 Results: Deployment at a distributed software-based research team  
6.4.1 Feature usage  
6.4.2 Canvas navigation  
6.4.3 Representations  
6.4.4 Collaborative work  
6.5 Design behaviors  
6.5.1 Design behaviors summary  
6.5.2 Kinds of sketches software designers produce  
6.5.3 How they use the sketches to navigate through a design problem  
6.5.4 How they collaborate on them  
6.6 Summary  

7 Discussion  
7.1 Strengths and weaknesses of each feature  
7.1.1 Scraps  
7.1.2 Palette  
7.1.3 Intentional interfaces  
7.1.4 Fading highlighter  
7.1.5 Summary  
7.2 Cognitive dimensions analysis  
7.2.1 Abstraction  
7.2.2 Closeness of mapping  
7.2.3 Consistency  
7.2.4 Diffuseness  
7.2.5 Error-proneness  
7.2.6 Hard mental operations  
7.2.7 Hidden dependencies  
7.2.8 Premature commitment  
7.2.9 Progressive evaluation  
7.2.10 Provisionality  
7.2.11 Role expressiveness  
7.2.12 Secondary notation  
7.2.13 Viscosity  
7.2.14 Visibility  
7.2.15 Summary  
7.3 Overall strengths and weaknesses  
7.3.1 Strengths  
7.3.2 Weaknesses  
7.4 Interviews with users  
7.5 Summary  

8 Related Work  
8.1 Tools that interpret  
8.2 Systems that support sketching activities
9 Conclusions and Future Work 240
   9.1 Directions for future work ........................................ 246
      9.1.1 Compositional notations ..................................... 246
      9.1.2 Informal analysis of scraps .................................. 248
      9.1.3 Contextualized sketching .................................... 250
      9.1.4 Design histories ............................................. 251

Bibliography 254

A The Prompt Provided to the Design Teams 261
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Whiteboard wall at a startup company.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Calico version one.</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>Calico version two.</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Diagram from a design session that includes UML diagrams, a map, and annotations.</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Diagram from a design session that was refined from a list into a UML diagram.</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Diagrams using impromptu notations.</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>The designer navigates between different types of diagrams and abstractions.</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>Two diagrams that designers used to discuss alternative approaches for modeling a traffic light.</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Designers sometimes use their sketches to mentally simulate their designs in use.</td>
<td>22</td>
</tr>
<tr>
<td>2.7</td>
<td>Designers juxtaposing two sketches by pointing.</td>
<td>23</td>
</tr>
<tr>
<td>2.8</td>
<td>While reviewing progress, designs may mark up their requirement lists.</td>
<td>24</td>
</tr>
<tr>
<td>2.9</td>
<td>Designers sometimes break into independent groups to work out solutions, before later synchronizing their insights.</td>
<td>25</td>
</tr>
<tr>
<td>2.10</td>
<td>The designer interleaves many design behaviors.</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Physical setup of Calico.</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Calico appearance at the start of a design session.</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Grid in use.</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Scraps.</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Calico architecture.</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Gestures in Calico for scraps and arrows.</td>
<td>47</td>
</tr>
<tr>
<td>4.7</td>
<td>Feature usage across the eight calico pairs.</td>
<td>53</td>
</tr>
<tr>
<td>4.8</td>
<td>Several impromptu notations emerged in the design sessions.</td>
<td>55</td>
</tr>
<tr>
<td>4.9</td>
<td>Examples of models in varying amounts of detail.</td>
<td>56</td>
</tr>
<tr>
<td>4.10</td>
<td>Lists were converted into representations of software components using scraps.</td>
<td>59</td>
</tr>
<tr>
<td>4.11</td>
<td>Representations of software components were broken up across several canvases.</td>
<td>60</td>
</tr>
<tr>
<td>4.12</td>
<td>Participants tended to focus their efforts towards a particular theme in each canvas.</td>
<td>60</td>
</tr>
<tr>
<td>4.13</td>
<td>How time was spent within and between the design activities.</td>
<td>66</td>
</tr>
<tr>
<td>4.14</td>
<td>Emergent pattern of transitions across both the Calico and whiteboard pairs.</td>
<td>68</td>
</tr>
</tbody>
</table>
5.1 Regular sketches can be transformed into selection scraps, which can further be made permanent by “pinning” the scrap. ................................. 83
5.2 Intentional Interfaces - Intention View. ................................. 85
5.3 Canvases are related to one another within intentional interfaces using tags. 86
5.4 Fading highlighter. ................................. 88
5.5 The Calico canvas interface, including the side panels and the drawing space. 96
5.6 Scrap functionality. ................................. 99
5.7 List-scrap automatically organize scraps into a vertical list. ................................. 102
5.8 Scraps can be added to palette for rapid reuse. ................................. 103
5.9 The Calico wall belonging to the SDCL group, which contains sketches that are several years old and are used by several group members regularly. .... 104
5.10 The cluster perspective of “Nick M.’s stuff”. ................................. 105
5.11 A diagram is refined and revised across several canvases, which are linked with “Continuation” and “Alternative” tags. ................................. 107
5.12 The tag panel appears in the upper-right upon entering a newly created canvas. 108
5.13 The breadcrumb bar in the upper-left enables users to navigate between canvases without using the wall or cluster perspectives. ................................. 109
5.14 Highlighter. ................................. 110
5.15 Scraps provide the building blocks to compose different representations, such as UML class diagrams and user interface mockups. ................................. 111
5.16 Figure 2.4 in Chapter 2 in Calico, in which UML class diagram components contain only as much detail as needed by the designer. ................................. 112
5.17 Designers may use scraps and connectors to refine their designs from plain sketches into representations that more closely approximate UML class diagrams. ................................. 113
5.18 The intentional interface view (shown far left) helps designers in moving between canvases of different perspectives, which may themselves contain multiple perspectives (center), or focus on just one perspective, such as requirements (far right). ................................. 113
5.19 The intentional interface view (shown far left) provides a birds eye view of the two alternative canvases to the right, one which is formal (center), and the other which is informal (right). ................................. 114
5.20 Intentional interfaces helps designers in moving between perspectives using tagging and organizing spaces. ................................. 114
5.21 The fading highlighter helps designers in performing mental simulations by allowing them to sketch over their diagrams without marking them. .... 115
5.22 Designers may work together synchronously by occupying the same canvas, or asynchronously by working in different canvases. ................................. 116
5.23 The fading highlighter helps designers in explaining their sketches to others. 117

6.1 The physical setup at the commercial OSS company. ................................. 123
6.2 A tutorial of the usage of Calico given to the interaction design group. ................................. 124
6.3 The physical setup of the research group. ................................. 125
6.4 Representations used by the OSS group in their first design session to design a user interface. ................................. 130
6.5 Representations used by the OSS group in their second design session in which they worked on representation that details the software architecture for handling different types of messages. 131
6.6 Representations used by the OSS group developer to refactor their code in the third reported design session. 132
6.7 Cluster view of canvases. The three major sessions were grouped by “chaining” canvases. 136
6.8 Various visual representations used by the interaction design group to create personas. *Note: content in sketches does not depict actual work by the interaction design group, but instead are fictional examples of types of representations that they used.* 149
6.9 Representations created by the research group during their one-week intense session. 154
6.10 Representations created by the onboarded researcher that were used to better understand the existing software system. 155
6.11 The OSS group sometimes spent as long as 30 minutes walking through the data flow of a diagram using the fading highlighter. 179
9.1 Imprecise values can be used to provide rapid feedback. 249
LIST OF TABLES

4.1 Calico features as they address the targeted subset of four design behaviors from Chapter 2. ................................................................. 36
4.2 Summary of design conversation categories. .................................................. 63
4.3 Correlations between sessions for total time spent (dark cells pertain only to Calico sessions, white cells pertain only to whiteboard sessions, and lightly shaded cells are the intersection of both). ........................................ 66
4.4 Correlations between sessions for transitions (dark cells pertain only to calico sessions, white cells pertain only to whiteboard sessions, and lightly shaded cells are the intersection of both). ........................................ 68
5.1 The set of design behaviors and how the features support them. ................. 88
6.1 The set of design behaviors and features that each group used to support them in the field evaluations. .................................................. 168
7.1 The strengths and weaknesses of each advanced feature. ............................. 188
7.2 Cognitive Dimensions Analysis of Calico. .................................................. 197
7.3 The set of design behaviors and the features that supported them. ............... 228
ACKNOWLEDGMENTS

My deepest gratitude to my advisor, André van der Hoek, for these many years of guidance and diligent patience in shaping me into an independent researcher. I thank André for sharing his wisdom, instilling an appreciation for due diligence in my work, and inspiring me to seek out new challenges. He has given me the freedom to explore new research ideas (even when we disagreed), helped me when I stumbled, and always pushed me to do my best.

I thank my other committee members: David Redmiles for his feedback and advice since the very early days of my work; Gary Olson for his insight and advice on my quantitative analysis; and Marian Petre for the many thought-provoking conversations that influenced much of my work.

I would like to express my appreciation to my mentors who have given me an industry perspective on research during my internships. In particular, Noi Sukaviriya and Harold Ossher from IBM Research.

To everyone in the research group, thank you for making my stay a memorable and wonderful experience. Thanks to Alex Baker, whose feedback and effort was key in putting this work into motion. Thank you to Thomas LaToza, whose work and guidance in quantitative analysis were valuable. Thank you to those who worked on the Calico project, including Mitch Dempsey, Dastyni Loksa, Michele Bonazza, Alfredo Motta, and Wayne Huang for their contributions. My thanks to everyone who has made my stay a joyful and memorable experience, particularly Gerald Bortis, Tiago Proena, Nilmox Moura, Nazia Chorwadwala, Eugen Nistor, Anita Sarma, Isabella Almeida da Silva, and Marcelo Alvim.

I would like to express my gratitude to the many collaborators in the research community who help me to develop my work. My thanks to Professor Cláudia Werner and Professor Leonardo Murta for their hospitality during my visits to Rio de Janeiro, and to Rodrigo Pereira for hosting me during my stay. Thank you to Werner Beuschel for the many supportive and constructive comments to help improve my work.

Thank you to my family for their years of love and encouragement, which has given me the confidence and strength to succeed in my endeavors.

I gratefully acknowledge the financial support provided by the NSF, ARCS, HP, and Microsoft.
CURRICULUM VITAE

Nicolas Francisco Mangano

EDUCATION

Doctor of Philosophy in Information and Computer Science 2013
University of California, Irvine

Masters in Information and Computer Science 2011
University of California, Irvine

Bachelor of Science in Information and Computer Science 2007
University of California, Irvine

Bachelor of Arts in Psychology 2007
University of California, Irvine
ABSTRACT OF THE DISSERTATION

Calico: An early-phase software design tool

By

Nicolas Francisco Mangano

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 2013

Professor André van der Hoek, Chair

When developers are faced with a design challenge, they often turn to the whiteboard. This is typical during the conceptual stages of software design, when no code is in existence yet. It may also happen when a significant code base has already been developed, for instance, to plan new functionality or discuss optimizing a key component.

While software design at the whiteboard has long received an interest from the research community in terms of providing automated support through electronic sketching tools, the tools to date fall short in that each supports only a small subset of the design behaviors that software designers exhibit while working at the whiteboard. Indeed, the key insight motivating the research presented in this dissertation is that designers do not “operate in” or apply just a small handful of such design behaviors, but flexibly switch among many of them, all the time. As such, tools that only support a few design behaviors at the expense of others are inadequate to fully support the rich activity of software design at the whiteboard.

In this dissertation, I present Calico, a distributed, collaborative sketch-based environment that supports a broad range of design behaviors. Calico offers four features: (1) scraps and connectors to manipulate content and build higher-order representations, (2) intentional interfaces to partition and organize the drawing space, (3) a palette to reuse content, and (4) a fading highlighter to make temporary marks on sketches. These features were designed based
on an extensive study of the literature on software designers at work, and yield a minimally-
invasive, yet rich environment for supporting design at the whiteboard that maintains the
fluid and sketchy feel of sketch-based design.

Calico was evaluated with designers in a controlled laboratory experiment, as well as in a
longer-term in-the-field deployment, which included two software teams in industry and one
research group. Findings show that, while there was a great amount of diversity in the
design approaches by the different groups, the features do support and indeed are used for
the design behaviors in actual work. Further, the designers reported that they sketched more
than they would have on the whiteboard, built and refined their own types of representations,
managed a growing number of sketches by organizing them into multi-page narratives, were
more likely to return to old sketches, and switched between synchronous and asynchronous
work during meetings.

The contributions of this dissertation are four-fold: (1) The establishment of design behaviors
as a necessary quality to support in informal design at the whiteboard. (2) A set of features to
support informal software design at the interactive whiteboard. (3) An extensive evaluation
of these features in a controlled laboratory setting and in the field with professional designers
using Calico to perform their everyday work. (4) The construction of an extensible, sketch-
based environment for early-phase software design.
Chapter 1

Introduction

Often when developers are faced with a design challenge, they will turn to the whiteboard. This is typical during the conceptual stages of software design, when no code is in existence yet, but may also happen when a significant code base has already been developed, for instance, to plan new functionality or discuss optimizing a key component. Design sessions at the whiteboard may even arise spontaneously, such as when a developer has to refactor some code or discuss how to best integrate a new feature.

Compared to the polished, precise, and typically detailed models software designers produce using modern software design tools, the content they create at the whiteboard consists of rough sketches and imprecise approximations of the design they have in mind, which are continuously modified and refined as part of the design activity [12]. The sketches on the whiteboard wall shown in Figure 1.1 illustrate this point. The sketches are the result of many design sessions at a startup software company. While the sketches clearly are unintelligible to those who were not present during the design sessions, they serve a crucial role for those who were: they were the vehicle for working through a complex design problem and making key decisions that defined the software to be developed.
Developers turn to the whiteboard for the flexibility and fluidity that it offers in the design experience [15]. On a whiteboard, developers can freely sketch, branch off to another part of the design problem, return to a previous part, erase some portion of their work, redraw it, and so on, all without the typical restrictions one might find in a traditional software design environment. They can focus on designing without worrying about particular notations, how to navigate within a tool, etc.

While a preferred medium for design, a significant disadvantage of the whiteboard is that it is a passive medium: it has no facilities that purposefully support the design process. Particularly, whatever is drawn or written remains static and cannot be manipulated, other than drawing over it or erasing it. This is a less than desirable situation, because it is known that software designers often wish to manipulate a design at hand in more advanced ways than merely adding or erasing content [22]. The whiteboard, thus, limits what they are able to do, or at least how efficient they might be at doing so.

The research community has acknowledged this problem and has contributed many approaches that rely on an electronic whiteboard to provide more advanced support (e.g., [14, 45, 36, 20, 16]). This support can be divided into two broad categories: (1) approaches
that focus on sketch recognition, and (2) approaches that focus on management of sketched content. Approaches in the first category, sketch recognition, attempt to interpret the strokes made by the user to turn them into formal objects. Early work offered a predefined visual vocabulary for converting sketches into formal objects, such as UML diagrams [14] or user interface mockups [45], with tools that provided feedback to the designer based on the rules of the formal notation they support. Later work made visual vocabularies expandable by users [36] and made using the tools more flexible by delaying interpretation until it was desired by the user [20], sometimes even while retaining a sketchy appearance [16].

Approaches in the second category, sketch management, help organize the potentially many and varied sketched artifacts that may be produced during meetings (Figure 1.1 is an illustration of this point). Early approaches provided access to a large number of whiteboards through a filmstrip [76], hyperlinks [77], or hierarchical perspectives [54]. Later work automated particular aspects of managing sketches by automatically grouping clusters of sketches in close spatial proximity [51], shrinking sketches when moved to the periphery [33], or using metaphors such as Post-It Notes to organize and relate sketches [42].

In examining these and other existing sketching tools, it is useful to consider their respective underlying motivations. In so doing, I observe that every sketch tool was designed to support a particular way of working at the whiteboard. For instance, Knight supports designers in refining initial rough sketches into more formal representations [20]. As another example, Flatland supports designers in creating many different diagrams by automatically clustering sketches and adding specialized behaviors to those clusters [51].

In this dissertation, I define these ways of working as design behaviors. More precisely, I define a design behavior as a recurrent, recognizable set of actions serving a single purpose within a design meeting. In the remainder, I scope these actions to pertain to the whiteboard, that is, focus on creating and modifying its content, navigating the content created, and collaborating in all of these. That is, two designers calling up a colleague is not a design
behavior in which I am interested. One designer refining a diagram drawn by another is.

Quite a few design behaviors have been identified in the literature, despite the fact that the study of software designers “in action” is still in its infancy. For example, in addition to refinement of sketches and supporting multiple different types of sketches, studies of designers at OOPSLA’s DesignFest found that software designers improvise their own notations and evolve their diagrams across many canvases [22]. As another example, in-the-field observations at software companies found that software designers deliberately switch among formalisms and use provisionality to engage in a dialog with incomplete ideas [63].

Broadly speaking, design behaviors that software designers perform can be be described as falling into three high level categories. The first is how they create and modify the kinds of things they draw, such as the different types of sketches they create and their improvisation of notations. The second is how they navigate those sketches, such as shifting focus between sketches of different types. The third is how they collaborate over their sketches, such as when designers switch from working together on a sketch synchronously to working asynchronously.

The key insight motivating our research is that software designers do not “operate in” or apply just one behavior for an entire design meeting. Rather, designers interleave design behaviors over the course of a design meeting, switching among them as they see fit to navigate a design problem and its potential solutions. For instance, a designer may first sketch two diagrams and juxtapose them side-by-side to evolve them in parallel, then record patterns of execution in one of the diagrams using an impromptu notation, and thereafter shift to a different aspect of the design problem altogether. Throughout, the designer fluidly exhibits a range of these different design behaviors, typically without an explicit trigger. Designers shift opportunistically, addressing the design at hand in the way they see most beneficial at that moment.

During a meeting, it is a natural choice for designers to limit themselves to a single tool
to support them. That tool is typically a whiteboard or paper [63], though in some cases it may be a computerized tool like the ones I have described above (e.g., SUMLOW [14], Knight [19], Flatland [51]). In the latter case, the choice of tool determines the behavior or small set of behaviors that are now supported, as designers will seldom move between tools during meetings, because of the high cost associated with switching, both in terms of the cognitive burden on the user to switch contexts and in terms of the effort required to import or manually copy the contents. The cost is simply too great and designers, thus, are stuck with support for at best a few of their behaviors as embedded in the tool they happen to be using.

What is desired is a tool that supports a broad range of behaviors and allows developers to use the features naturally when they need them. Creating such a tool, however, is a non-trivial exercise. Simply picking up functionality from one tool and dropping it in another, and doing this repeatedly to support a multitude of behaviors, leads to tools that are highly disjoint. It is unclear, for instance, what it would mean for a tool to have available both multiple canvases in a filmstrip and functionality that automatically makes room on the current board? As another example, a tool that automatically recognize sketches and also supports emergent notations is equally difficult to envision as being conceptually clear to its users. Existing solutions do not necessarily stack their functionality gracefully, and the approach taken by one tool may collide with the support provided by another.

This dissertation explicitly addresses the interleaving behaviors that designers exhibit during software design at the whiteboard by taking a step back, examining a collection of behaviors, and contributing a new tool that is designed from the ground up to support this collection of behaviors with a small set of conceptually coherent functionalities. In order to put this into practice, I first built a basic sketching interface so that, as a baseline, designers may sketch and perform the same activities as they normally would on a whiteboard, but in a digital medium. Building on this foundation, I incrementally introduced features that support one
or more design behaviors, yet minimally obstruct the support for other behaviors.

This approach was realized by the construction of a software tool called Calico. In this dissertation, I describe two versions of the tool, the first of which is an initial exploration in supporting a subset of the design behaviors with a sketch-based tool, and the second of which significantly iterates on the first to support the complete set of design behaviors I set out to support.

The first version of Calico supported an initial subset of the design behaviors with three main features. Figure 1.2 depicts the interface for this version of Calico. The first feature, scraps, supported the kinds of sketches that designers created by providing a mechanism to create representations for box-and-arrow type diagrams and for manipulating sketches. The second feature, the grid, supported designers in navigating between sketches by providing a grid-layout of all canvases. Depicted in Figure 1.2(b), users can partition their work over the canvases in the grid, and move the resulting canvases around to organize their work. The third feature, the palette, allowed users to save scraps (depicted on the right of Figure 1.2(a)), and reuse them by dragging them back onto the canvas. Chapter 3 explains these features in greater detail.
Calico version one was evaluated in a controlled laboratory study in which I compared the design activity of computer science graduate students engaged in a challenging software design problem using Calico against those using a regular whiteboard. I presented all participants with a prompt to design an educational traffic simulator and asked them to either design it using either Calico or the whiteboard. I found that the same design behaviors performed at the whiteboard were also performed in Calico, but in a slightly different manner. The participants in the Calico groups did use Calico’s advanced feature set to perform the design behaviors, and reported that they found the features useful. However, the participants rarely used the palette. The design conversation of the groups were also analyzed using protocol analysis, which was done by breaking down the sessions into segmented phrases belonging to different categories of design. The breakdown of the categories demonstrated a very high correlation with reports from past research of how software design was performed at actual software companies, providing evidence that the design as it happened in our study matched that of design “in the wild”. Lastly, the groups reported that the limitation of the interactive whiteboard used to only accept a single person drawing at a time hindered their ability to perform parallel work.

Following our experiences with Calico Version One, I rebuilt the tool from the ground up in Calico version two. The features in the second version of Calico significantly iterate on those in the first version to support the whole set of fourteen design behaviors I identified as crucial to designers at the whiteboard. This version is depicted in Figure 1.3. The scraps feature was significantly revised based on user feedback to better support the designers in the kinds of sketches designers create. The grid feature was replaced with intentional interfaces, depicted in Figure 1.3(b), in order to better support designers in navigating among projects and the canvases capturing each project. In our experience with the grid, I saw that users typically dedicated entire rows or columns to a single topic. That behavior inspired intentional interfaces, in which users may declare their intent, such as “exploring an alternative”, when entering a new canvas. In the intentional interfaces view, canvases
are laid out in a circle, and a set of canvases focusing on a particular topic can extend outwards radially (Figure 1.3(b)). Relationships between associated canvases are explicitly captured as well using light-weight tagging. Lastly, the software architecture of Calico itself was rebuilt using a client-server architecture in order to allow designers on different machines to collaborate on the same sketch. Chapter 5 explains these features, as well as others, in more detail.

Calico Version Two was evaluated by deploying it at three field sites, including a commercial open source software company, an interaction design company, and a geographically distributed research group. All groups had access to a large electronic whiteboard running a Calico client, as well as pen-based tablets to connect to a shared Calico server. Members of each respective group conducted extensive design sessions using Calico to perform their own work within their respective organization. I collected and reviewed usage logs from the Calico server, and also conducted semi-structured face-to-face interviews. From this data, I evaluated the use of Calico’s features in supporting the representations they created, their strategies for navigating between sketches, and how they collaborated on their sketches.
Overall, the field evaluations gave insight into how Calico supports design in a real world setting. Usage at the commercial open source software company provided insight in how Calico supported a group engaged in detailed coding, with the group at large and individuals using Calico multiple times, and also personal sessions over both software components and software code. Usage at the interaction design group provided insight into how Calico helped them create personas and storyboards, for example, by organizing pictures of those they interviewed in one- and two-dimensional plots. Lastly, usage by the research group provided insight into how Calico supported a set of distributed researchers in a long-term collaboration. Their use included the development of a software system over a period of several months, with intensive sessions during which they designed process flow diagrams to model the behavior of their software. Overall, the four features collectively supported all fourteen of the design behaviors both in ways that I did and did not expect.

1.1 Thesis structure

This thesis is organized into nine chapters. The remaining chapters are structured as follows:

Chapter 2 - Motivation. This chapter introduces the definition of design behaviors in detail and presents the grand set of design behaviors that I aim to support. The research question of this thesis is also introduced here.

Chapter 3 - Research Question. We pose the topic that this dissertation will address, layout a method by which I approach this topic, and how I will evaluate this approach.

Chapter 4 - Calico Version One. I describe the approach and features of the first version of Calico. This chapter introduces the first concepts of scraps, the grid, as well as the palette. The approach is evaluated in a comparative study between Calico and the regular whiteboard, both with respect to the design behaviors performed as well as the design
conversations that took place.

Chapter 5 - Calico Version Two. The second and final version of Calico is introduced in this chapter. The new features are introduced, including the revised interaction for scraps, the intentional interfaces features that replaces the grid, the distributed nature of the architecture, as well as other features targeted at supporting the full set of design behaviors.

Chapter 6 - Evaluation. Here I present our evaluation of the final version of Calico “in the wild” at a commercial open source company, an interaction design company, and a distributed research group. A comprehensive evaluation was performed across three field sites, examining the use of features to support the representations used, the navigation among the sketches, and the collaboration patterns.

Chapter 7 - Discussion. In this chapter, I take a step back and put my work in a broader context. I discuss the implications of my findings, reconnect my work to the broader design literature, and provide some findings and observations that are outside the evaluative framework of the design behaviors, but are worthwhile to highlight nonetheless.

Chapter 8 - Related Work. I review the existing literature of tools that support software design on an electronic whiteboard, covering both tools that support sketch recognition and those that support the management of sketches.

Chapter 9 - Conclusions and Future Work. This chapter summarizes the contributions of my dissertation, and also provides some concluding remarks, and offers an outlook at future work. I particularly suggest the following future avenues of research: (1) the concept of compositional notations, which builds on scraps to provide more targeted support for software design notations, (2) informal analysis of scraps in order to provide intelligent feedback on designs in progress such as critiquing the performance of an activity diagram, (3) contextualized sketching, which may import artifacts from an existing code repository to help with design discussions in Calico, and (4) design histories, which may help with
reflecting on design activities performed.
Chapter 2

Motivation

2.1 Design behaviors

Before introducing the design behaviors that I aim to address in my research, it is useful to firmly define a design behavior. I define a design behavior as a recurrent, recognizable set of actions serving a single purpose within a design meeting. This definition recognizes the breadth of activities that may occur during a design meeting, from phoning a colleague, to using sticky notes to brainstorm, to sketching some concepts. For purposes of this dissertation, however, we scope our investigation to design behaviors that occur at the whiteboard, and specifically those pertaining to creating and modifying whiteboard content, navigating the content created, and collaborating in doing so.

I make four important observations regarding this definition:

- **recurrent** – A design behavior can be observed to happen consistently across many design meetings and across many designers. Performing a particular action once in a meeting, such as drawing a diagram of an intersection, does not make ‘intersection
drawing' a design behavior. Repeated behavior, particularly as applied to different types of diagrams, across multiple design meetings is required.

- **recognizable** – A design behavior stands out as a coherent set of actions within an overall design meeting. The actions clearly belong together and can be distinguished from other groups of actions.

- **set** – A design behavior necessarily unfolds over time with multiple actions that a designer undertakes. A single stroke or gesture does not qualify.

- **serving a single purpose** – A design behavior has a purpose in the overall exploration of a design problem and its potential solutions. The set of actions contribute to furthering the exploration.

With this definition in hand, I examined both the software design literature and the broader design literature for design behaviors. What I found was that the general design literature is more mature than the software design literature, in that it has identified quite a few design behaviors that span across different design disciplines. For instance, designers across building architecture, engineering, and product design use constraints to guide their design thinking [17]. As another example, designers in multiple fields use visual similarity between their sketches and the final product to help them envision the eventual physical artifact [25].

The software design literature is only now beginning to catch up to the topic of design behaviors, with the emergence of studies that are beginning to look at software designers “in action” (e.g., [6, 15, 22, 63]). These studies, thus far, confirm the behaviors seen in other design disciplines, but at the same time do not confirm all of them yet, simply because the number of studies remains small. In the below, I include a subset of the design behaviors found in the general design literature, quite a few of which have already been confirmed in the software design literature (i.e., Behaviors 1, 2, 3, 4, 6, 7, 8, 9, 10, and 13) and some of which I believe will be confirmed in future (i.e., Behaviors 5, 11, 12, and 14). I feel confident
in making this assumption because of my own informal observations of software designers in action, particularly in studying the videos of the SPSD 2010 workshop [6]. While I have not performed full studies of those videos expressly for the purpose of corroborating the general design literature, informally I have seen multiple instances of all of the behaviors I discuss in the below.

I separate the fourteen behaviors I intend to support with Calico into three categories:

1. Kinds of sketches software designers produce

2. How they use the sketches to navigate through a design problem

3. How they collaborate on them

Each of these categories I detail below.

### 2.2 Kinds of sketches software designers produce

The first category deals with what the designers draw. The sketches that designers make at the whiteboard are typically not the goal in and of themselves and, as such, the types of sketches that designers make will vary depending on their current design activity. Sometimes, the sketches are used to help the designer in their thinking, by externalizing their ideas and thoughts onto the whiteboard [46]. Other times, the whiteboard is simply a medium to explain a thought, idea, or design in progress. The developer is not using it for problem solving, but instead to communicate information to a listener or collaborator [26]. Because of these different purposes, what designers draw is dependent on what they work on at what point during a design meeting. This leads to the following design behaviors.
1. *They draw different kinds of diagrams.* In order to explore a design problem, software designers sketch many different types of diagrams, often within the same canvas [5,17]. They may sketch, for instance, entities and relationships, interface mockups, scenarios, architectures, and other kinds of diagrams [15]. The freedom to sketch multiple kinds of diagrams in the same space is fundamental to supporting the exploration of the design space, as it enables designers to explore an issue from different angles, at different levels of abstraction, or even in different ways altogether. The sketches in Figure 2.1, which contains a snapshot of the whiteboard from a design session in which two software designers are designing an intersection in a traffic simulator, present such an example. On the right, the designers used the sketch of a traffic intersection to help them in working through and refining the UML model of some of the entities in the simulation on the left. In turn, their work on the entities lead to revisions to the sketch of the intersection on the right. This behavior of working with multiple representations aligns well with the observed phenomenon that tools which restrict designers to use one notation hinder the design exploration and lead to fewer alternatives being considered [74].

2. *They produce sketches that draw what they need, and no more.* Of the many sketches that software designers create, few are drawn in full detail. Software designers typically can get what they need from a quick and incomplete sketch, e.g., a barebones user interface or boxes-and-arrows sketch [80]. The benefit of a low-detail sketch is that it can be created quickly, and modified easily, giving the designer more rapid feedback [15, 63]. Further, providing too much structure too soon can create unconscious barriers to change, resulting in a less exploratory and a less broad search for candidate solutions [82]. This behavior further breaks down into two parts:

(a) *They only draw what they need with respect to the design at hand.* Low-detail sketches tend to incorporate relevant information and omit the irrelevant [78],
including only as much detail as necessary to advance the designers’ thinking. When talking, they will only draw what they need to reinforce what they want to communicate [63]. When thinking, they will only draw the details relevant to the immediate issue to help them reason [22]. For example, the elements in the left of Figure 2.1 differ in the amount of detail they contain, where some contain data attributes and others contain only a name.

(b) They use only those notational conventions that suit drawing what they need. Sketches only include as much notational convention as the designer needs in a given situation [63]. For instance, if a sketch can express an idea using only boxes-and-arrows, then no more will be drawn, but if a sketch must represent a hierarchical relation, then a richer array of arrows will be present, typically following the convention of an existing formal notation.

3. Over time, they refine and evolve their sketches. The level of detail that designers want in their sketches varies over time. Early on, they may need very little, but later
they may need much more as they expand on their ideas [60]. Figure 2.2 presents an example of such a refinement, which details the evolution of the diagrams from Figure 2.1 from a relatively simple list, to a complex box-and-arrow diagram. As designers discuss and work through an idea, they will evolve their sketches with additional details to capture their decisions. In general, as part of this refinement process, the sketches will contain more visual precision, and the designer will rework the design to fix any inconsistencies [20]. This behavior, too, breaks down into two parts:

(a) *They detail their sketches with increasing notational convention.* As a designers understanding of the design space matures, so does the representation that they use. As I already discussed, while the designer is aware of the full expressive powers of formal notations, they only borrow from those notations what they need at the time. However, as the designer progresses and the decisions firm up and additional, less-critical decisions are made, they tend to use more and more of the formal notational convention to represent their commitment to the chosen direction [59]. The formal elements in Figure 2.2 illustrate the decisions captured by the designers, where some arrows contain information about cardinality and some elements are contained in a bounding box to reflect their status as an entity in the model. Note that this move towards a more formal notation is not a strictly uniform activity, as different parts of the design may exist at different levels of maturity [63].

(b) *They appropriate a sketch in one notational convention into another notational convention.* Refinement of sketches does not always mean refinement toward and in a single notational convention. Sometimes, designers appropriate one kind of diagram into another [22]. For example, the designers initially created the data model in Figure 2.2 as a list, then later added boxes, then boxes with arrows, and finally evolved the sketch into a UML class diagram. The designer in all likelihood did not plan this, but in working out their design in place, they re-appropriated the
Figure 2.2: Diagram from a design session that was refined from a list into a UML diagram.

sketch to suit their needs [48]. They use what is readily available over re-creating a similar diagram, especially if they do not anticipate needing the original diagram later.

4. *They use impromptu notations.* Designers do not exclusively work with the notational convention they know (e.g., UML, ER, etc.), but also, at times, will improvise in the moment. The deviations that they make from standard notations, such as annotating UML diagrams with custom symbols, are deliberate additions that break convention to capture insights before an idea is forgotten. Beyond such annotations and minor deviations, developers also will sometimes adapt wholly new notations on the fly. These often relate to the problem domain that they are explaining, since few domain-specific notations exist, but shorthand is still needed to support the design process [22]. For example, Figure 2.3(a) uses freeform text to annotate what looks like a stripped down UML diagram rather than standard UML notation to include additional information. Further, Figure 2.3(b) uses rectangular boxes with circles inside to represent a queue of
cars waiting for a traffic light, circles with edges to represent traffic lights that govern each queue of cars, and a clock symbol to signify that events in the intersection occur according to a timer component. The designers created these notations on the spot to capture the ideas in their conversation.

2.3 How they use the sketches to navigate through a design problem

The second category pertains to how designers use sketches to navigate the design space. While designers may create many different sketches over the course of a design meeting that vary in detail, notation, and what they represent, there is typically a thread of thought that relates the sketches and the ideas represented in them to one another.

5. *They move from one perspective to another.* Software designers create many sketches through which they shift their focus between perspectives. The designer in Figure 2.4,
for example, is working on the user interface component for a map, and may navigate to the two different views of the intersection to his left, or explore the UML model of its data model to the far left. The designer uses each new perspective to better understand how the parts of a design fit into the whole, asking questions such as: “[what] if we look at it like this, from this angle, it fits together like this” [63]. Each perspective presents a new way of looking at the same design, and what may be subtle in one perspective, may be more pronounced and easier to understand in another.

Figure 2.4: The designer navigates between different types of diagrams and abstractions.

6. They move from one alternative to another. In a sufficiently complex software design task, a software designer will generate sketches of competing solutions before committing to a particular choice [84]. Expressing alternatives as sketches rather than simply mentioning them aloud allows designers to manage their focus and more effectively explore alternative solutions [50]. Once created, designers can compare alternatives and weigh their trade-offs [12] (see behavior 5). They may shift their attention back and forth between alternatives and adopt ideas proposed in one into another, or synthesize the ideas of several alternatives into an entirely new alternative [40]. For example, the sketches in Figure 2.5 depict two approaches that could be used to model a traffic light, where the left sketch models the light as a single signal with a red, yellow, green, and left turn signal, and the two right sketches use two lights that are separate and make the left turn signal independent from the regular traffic symbol. Placing these diagrams
side-by-side helps designers in visualizing and discussing the differences between the two.

Figure 2.5: Two diagrams that designers used to discuss alternative approaches for modeling a traffic light.

7. **They move from one level of abstraction to another.** Software designers move between different levels of abstraction, either by “diving into” parts of their design to explore them in more detail or by shifting “back up” to the higher-level representation. For example, the designer in Figure 2.4 has navigated to the sketch of a map, which presents a zoomed-out view of the two intersections to his left. This shift in abstraction happens often in software design, as many of its notations are hierarchical in nature. A software architect may shift their focus from working out how software components interact with each other to choosing a component and working out how it functions, perhaps by drawing its internal architecture and diving in even further. This behavior typically leads to a multitude of sketches that together consider different abstractions simultaneously [63]. Many scenarios requiring shifts of abstraction have been documented, including the design of user interfaces [18], web pages [79], and so on.

8. **They perform mental simulations.** Software designers use mental walkthroughs to gain insight into the consequences of their design [83]. They may need to understand how information flows among components, or inspect their design by mimicking how an end user would interact with it. The software designers ‘interrogate’ their design by
testing it against hypothetical inputs and scenarios, often marking over their existing sketch while simulating. For example, while discussing the logic cars use in moving through intersections, the designer in Figure 2.6 runs his finger along the path of the map (shown by the white arrow) to mentally simulate the path that a hypothetical car may take, and verbally walks through the logic the car uses while doing so. Through these mental exercises, the designers can bring to light their implicit assumptions and expose flaws in the design [63].

Figure 2.6: Designers sometimes use their sketches to mentally simulate their designs in use.

9. *They juxtapose sketches.* In order to compare and contrast ideas, software designers will often juxtapose sketches across perspectives, alternatives, and abstractions [63]. A class diagram may be examined in parallel with a sequence diagram to aid the designer in determining how a message is passed between components. As another example, the designer in Figure 2.7 points to both a data model and a map to understand how
a car object is passed between components as it travels through an intersection. The juxtaposed diagrams help the designer in reasoning how the design might work, using the knowledge gained from one diagram to help identify the omissions or mistakes in another, as well as any inconsistencies between them [63].

![Designers juxtaposing two sketches by pointing.](image)

Figure 2.7: Designers juxtaposing two sketches by pointing.

10. They review their progress. Not all time spent during design is dedicated towards producing new content or verifying whether the design does what the designer intends it to do. At some point, the designers must take stock of what they have done. They momentarily take a step back, away from the design, and consider the progress that they have made and what they have yet to do [48]. They may return to the problem statement or list of requirements and mark off everything they have done to address it, they may generate a new list of issues that they further need to address, or simply just talk amongst themselves, to assess where they are. As an illustration, the designers from Figure 2.8 circled a particular subset of their requirements list and explicitly related the items related to that requirement that may have issues that are intertwined with another issue.
11. *They retreat to previous ideas.* Periodically, designers may reach a stopping point in the exploration of their current set of sketches, such as when they become stuck or simply have exhausted an alternative. They then may choose to return to a previous state of the design (and its sketches) to start anew [83]. For example, an abandoned proposal for a time-based architecture may become a more lucrative option if an event-based architecture proves too costly in system memory usage. In returning to past ideas, the designer may bring new insights and a matured understanding from the exploration they just exhausted, which they can use to explore the past ideas further.

### 2.4 How they collaborate on them

Software design is a highly collaborative activity, especially at the whiteboard where sketching and design exploration is almost always performed in collaboration with others. The behaviors in this section result from the collaborative aspects of working toward a single
12. *They switch between synchronous and asynchronous work.* While much of the work that takes place at the whiteboard is typically synchronous, with all participants focusing on a single aspect of the design they are discussing, it is known that designers occasionally break away to explore an idea independently while the others continue with the main discussion [21]. This typically occurs when a sudden inspiration strikes, or when a designer wishes to develop a counterexample or alternative to what is being discussed now. The designers in Figure 2.9, for example, have split to independently work on different aspects of the design. The female on the left is exploring exactly how the timing of an intersection would work using a line graph, whereas the male on the right is working through a UML representation that explores the persistent aspects of the timing mechanism.

![Figure 2.9: Designers sometimes break into independent groups to work out solutions, before later synchronizing their insights.](image)

13. *They explain their sketches to each other.* After any independent work takes place, and even in cases where one designer is drawing on behalf of the group and “has the floor”, the designers must synchronize their mental models of the state of the design [22]. This behavior relates to Behavior 9, but represents its collaborative version. They will need to verbalize their mental simulations to explain the consequences of a particular
choice, clarify the meaning of a sketch, or even simply explain their assumptions or inspiration. While explaining, they may sketch on top of the existing diagram, using their marks to guide attention or add detail, or simply gesture over the diagram if they do not want to edit the sketches.

14. *They bring their work together.* Sometimes, as a result of asynchronous work, the designers need to integrate their ideas from separate sketches into a unified design. This may involve bringing parts of a sketch over, creating a new sketch that integrates both, or sometimes working on a third alternative that combines the best aspects of each, but requires a different underlying approach to make that work [22].

### 2.5 Summary

Taken together, these fourteen design behaviors capture ways in which designers work at the whiteboard, ways that we seek to support in my work. Designers continuously exhibit these behaviors, making choices such as how much or how little they sketch out a design, how they navigate their focus from one sketch to the next, or how they collaborate over these sketches.

These design behaviors, of course, never stand in isolation. Quite the contrary, any design meeting consists of an interleaved sequence of them. In explaining a design, a designer may navigate between different perspectives, alternatives or abstractions, and create and refine diagrams of different detail in the process. For example, the designer in Figure 2.10 explains an idea to her design partner (Behavior 12) by mentally simulating a car moving along the map (Behavior 8), and points to different perspectives (Behavior 5) during the mental simulation to show how it affects various parts of the program.

Given that these design behaviors occur naturally at the whiteboard, I aim to provide support for all fourteen design behaviors in order to improve the capability of the designer at the
Figure 2.10: The designer interleaves many design behaviors.

whiteboard. The design behaviors must be equally supported, where the support for any particular one should not impede the designer’s ability to perform another. As such, I must take an incremental approach in exploring support for these behaviors to ensure any added features meet this criterion.
Chapter 3

Research Question

Now that I have introduced this set of design behaviors, I return to what it would mean to build support for it. I first recognize that there is a spectrum of ways in which the design behaviors could be supported. On one end of the spectrum lies the whiteboard itself, a minimally intrusive, informal medium that supports only drawing and erasing of content. The naturally occurring behaviors are permitted, but not necessarily supported, which is the basic problem that this research is trying to address. On the other end of the spectrum is the formal design tool, a highly structured, heavy-weight environment with hosts of explicit features. Examples of such tools are Rational Rose [68] and ArgoUML [70]. Theoretically, they support a number of the behaviors, such as, for instance, Behavior 5 by providing multiple views on the same model, or Behavior 6 by maintaining many projects. They, however, do not nearly support all, and the ones that they do are not nearly as fluidly supported as necessary.

Between the informal whiteboard and the formal tool lies a range of possible approaches that provide different blends of the strengths of each. This leads to my research question:

What minimally invasive, coherent set of features can be designed that is sufficient to
effectively support these behaviors?

I am particularly interested in exploring this question in the context of an electronic whiteboard, that is: what kind of support can we implement that transforms the interactive whiteboard from a place where software designers just sketch like they normally would to a place where their design behaviors are explicitly enabled through the functionality of the tool they are using? Most tools, today, support one behavior or at best a few.

In presenting this research question, it is useful to examine its precise phrasing:

- **minimally invasive** – I seek to build support that does not completely abandon the whiteboard experience. People go to the whiteboard for its fluidity and flexibility. Any solution that I design must preserve the feel of the traditional whiteboard as much as possible.

- **coherent set of features** – I wish to arrive at a set of features that build on each other using a unified set of design principles and metaphors. Building an isolated feature for every behavior would not satisfy my research goal.

- **sufficient to support all of these behaviors** – Each behavior should in some way be supported by the overall set of features, and support for any particular behavior should not come at the cost of another.

Having put forward this research challenge, I must analyze the resulting effect of the support for the design behaviors as well. This is a difficult question to answer, particularly since the literature to date has documented that these design behaviors take place, but has not yet fully articulated their effects on the design process and design product. For instance, switching among perspectives (Behavior 5) seems to have a positive effect on the eventual design [5], and, in certain cases, it has been documented that the consideration of multiple alternatives
(Behavior 6) also seems to lead to a better design [12]. However, even these studies are hard pressed to provide absolute answers. The first study does not examine the optimal length of time a perspective should be explored: is thirty seconds too short, thirty minutes too long, or what is the general distribution? Similarly, the second study does not talk about the number of alternatives: is three sufficient, should twenty be explored, or does it depend on the design problem in some way? These are questions that cannot be objectively answered at this time.

For this reason, my evaluation must be exploratory. An obvious evaluation would examine factors such as the frequency of design behaviors, time spent working in each behavior, and how my solution impacts the interleaving of the design behaviors. A more comprehensive evaluation would attempt to correlate the appearance of these design behaviors with the quality of the process and the quality of the design by, for instance, relying on outside experts to rate the designs that are produced and correlating this quality with the frequency, interleaving, etc. of the design behaviors. While this would possibly lead to a conclusive statement with respect to the impact of my solution, it would be very challenging to perform this evaluation given that a typical design exhibits complex interleavings of many design behaviors, making achieving any statistical relevance difficult.

My evaluation, therefore, will take a qualitative approach. By observing Calico in practice and by interviewing users of the tool, I will analyze the usage of Calico within the scope of the design behaviors and analyze the impact of the features on those design behaviors. Additionally, while the observations will likely pertain to the existing set of fourteen behaviors, I of course do not rule out that new behaviors may emerge, as enabled by the features of the software, that might be worthy of study.

As such, the summary of the approach in the remainder of this dissertation is that I will describe Calico (two different versions) as well as the result of a qualitative evaluation of Calico in use at several local companies.
Chapter 4

Calico Version One

Calico Version One is the first complete prototype that was built and evaluated to directly support the design behaviors of Chapter 2. The prototype was built toward testing the feasibility of my approach. That is, 1) can I successfully build on top of the existing whiteboard experience with features that support the design behaviors, and 2) can I do so without obstructing the design activity itself.

I target only a subset of the design behaviors with the first version of Calico. In particular, this first version supports design behaviors 1, 2, 3, and 5, as summarized here:

- Design behavior 1: they draw different kinds of diagrams.
- Design behavior 2: they produce sketches that draw what they need, and no more.
- Design behavior 3: they refine and evolve their sketches over time.
- Design behavior 5: they move from one perspective to another.

Our express purpose with this first version is to evaluate the viability of the overall idea of supporting design behaviors with tool functionality at the whiteboard. The features are
therefore limited, as compared to the full version of Calico described in Chapter 5, and the evaluation focuses on building a basic understanding of the effect the features. Specifically, the evaluation presented in Section 4.3 looks at how often Calico’s advanced features were used, whether use of the features coincided with design behaviors, and whether the resulting design process – as judged by the nature of the design conversations – resembles that of design at the regular whiteboard.

In designing the first features of Calico, I avoided those which would inhibit the flexibility and fluidity of the whiteboard. Features from traditional point-and-click interfaces such as different modes, extensive menus, or a selection lasso were not included because such functionality is likely to be frustrating on a pen-based interface [1] and may disrupt the fluidity of the whiteboard experience. Further, rather than including support by providing sketching support for an existing software design tool (such as with Marama Sketch [32]), I took the approach of building a basic sketching tool first, and adding features incrementally to this foundation.

Through an iterative process of design, experimentation, and careful assessment, I arrived at four features. The features are: (1) a grid to manage multiple canvases in the workspace, (2) scraps that enable advanced manipulation of sketched content, (3) a palette to save and reuse scraps, and (4) a gesture-based interaction scheme to tie all features to stylus-based input.

The remainder of this Chapter is organized as follows. Section 4.1 presents Calico and how it addresses the stated goals. Section 4.2 presents my methods for evaluating Calico. My results are presented in Section 4.3. Section 4.4 includes a discussion of the results and Section 4.5 reviews the threats to validity within the study. Lastly, Section 4.6 summarizes the chapter and its contributions.
4.1 Features

In this section, I present the four novel features of Calico Version One, describe them in detail through various examples, and carefully relate them back to the subset of supported design behaviors I target in Chapter 2. As a guide, Table 4.1 shows a mapping between the subset of four behaviors targeted design behaviors and the main features of Calico.

Before I discuss the features, it should be noted that Calico is intended to be used on an electronic whiteboard (or Tablet PC, in the case of sole person use), and is optimized to take its input from a digital stylus, not a mouse. The designer stands in front of the electronic whiteboard and uses the digital stylus to draw, write, and control Calico (see Figure 4.1). The nature of this interaction shaped the design of Calico, particularly the mechanisms with which it enables designers to manipulate a design.

Note that all of the examples I use in this section draw from the evaluation design sessions I describe in Section 4.2.

4.1.1 Basic features

Figure 4.2 presents Calico as it first appears when a developer starts it. Users can immediately draw or write, without needing to enter any mode or selecting a widget to create new content. Just as on a standard whiteboard, they make any marks they wish, anywhere, in any shape. The drawing canvas has just a few visible widgets to maintain the appearance of a standard whiteboard. Seven colored buttons at the bottom left allow a user to change colors. The colors were carefully chosen to be in direct correspondence to the standard pen colors available on a regular whiteboard.

In addition to changing colors, Calico provides a small handful of features familiar to com-
Figure 4.1: Physical setup of Calico.

Figure 4.2: Calico appearance at the start of a design session.
Table 4.1: Calico features as they address the targeted subset of four design behaviors from Chapter 2.

<table>
<thead>
<tr>
<th>Design behavior</th>
<th>Feature</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design behavior 1: Use different kinds of diagrams</td>
<td>Scraps</td>
<td>Scraps maintain the shape of the objects that are drawn, and thus, together with rapid manipulation and creation of arrows, can emulate different notations and diagrams</td>
</tr>
<tr>
<td></td>
<td>Palette</td>
<td>Scraps can be copied into the palette from which they can later be reused</td>
</tr>
<tr>
<td>Design behavior 2: Draw only what they need</td>
<td>Scraps</td>
<td>Scraps do not force the user to draw more than they want to; they require no additional manipulations other than simply drawing the scrap. The user, thus, can draw as little or as much of a notation as they wish.</td>
</tr>
<tr>
<td>Design behavior 3: Refine and evolve sketches over time</td>
<td>Grid</td>
<td>Abstract sketches can be partitioned and evolved across several canvases, which can be meaningfully organized on the grid</td>
</tr>
<tr>
<td></td>
<td>Scraps</td>
<td>Strokes can be transformed from plain sketches to first order objects by making them into scraps, and then related to one another with arrows</td>
</tr>
<tr>
<td>Design behavior 5: Navigate between different perspectives</td>
<td>Grid</td>
<td>Sketching effort can be partitioned across multiple canvases Tab permits quick shifting between, and copying of, canvases</td>
</tr>
</tbody>
</table>
puter users. Users can pan a canvas, save and load designs, export sketches as images for use in other programs, as well as undo and redo actions. The undo and redo actions are globally stored, so if a previous action was performed on another canvas, the user’s view will shift to that canvas and the action will be undone. Also, while Calico does not recognize or formalize shapes drawn by the user, Calico does compensate for the typically low polling rate of touch-based hardware by approximating the curve of the user’s strokes using Bézier curves. Calico does not support text entry, relying on users to simply write on the board in order to avoid unwanted breaks in concentration due to the need to switch to an extraneous keyboard. While Calico will run on any touch-based device with Windows, Linux, or Mac OS (see Section 4.1.5), all of my experimentation was performed on a board with a projector resolution of 1024x768.

4.1.2 Grid

Calico’s first novel feature is the grid, which is my answer to the designer’s need to not only easily navigate between perspectives (Behavior 5), but to also be able to keep track of their design at hand. Shown in Figure 4.3, the grid provides the designer with a bird’s eye perspective of the design process as it has unfolded thus far by showing multiple canvases at once. By tapping on a canvas on the grid, the developer enters that particular canvas, where they can create new sketches or modify any sketches already in existence. By tapping the grid icon (see top right of Figure 4.2), the designer returns to the grid.

The grid naturally leads to a partitioning of design effort, while at the same time supporting straightforward movement between the emerging parts of this overall effort. For instance, a designer can make a to-do list in one canvas, use other canvases to work out each item on the to-do list in isolation, and return to the to-do list periodically to check and mark progress. As another example, the designer could choose to examine a design problem from various
perspectives or at varying levels of detail in different canvases. The grid becomes the record keeper of this exploration, allowing rapid shifting from one perspective or level of detail to another.

Previous tools have used the filmstrip metaphor to support multiple canvases, with a scrollbar at the bottom, typically in reverse order of manipulation [75]. With frequent updating of the content of different canvases, this leads to a volatile representation in terms of the order of the canvases in the filmstrip, making it difficult to navigate and easily shift between perspectives. The grid advances on the filmstrip by transitioning from a time-based metaphor to a spatial metaphor. The grid keeps all of the canvases in a constant location, which means that a user can not only easily locate individual sketches based on where they exist in the grid relative to each other, but also navigate to adjacent canvases without first having to switch to the grid. They can move left, right, up, or down one canvas at a time by using the tabs that are located in the middle of each edge of the canvas (see Figure 4.2). Tapping the white part of the tab enacts the move to the adjacent canvas in that direction.

Tabs also support a different form of navigating between sketches: exploration of alternatives.
On a standard whiteboard, it is difficult to fork ideas. One has to manually replicate a sketch elsewhere before modifying it. This is clearly undesirable and either leads to less exploration or direct manipulation of the sketch in question, the latter destroying the original from which the departure is being made and making it difficult to return to a previous state. By tapping the grey half of a tab, a canvas’ content is copied in its entirety to the corresponding adjacent canvas, which in turn is placed into focus. With a single “click”, thus, the designer is provided with a fresh copy of an entire canvas, which they can then further explore, refine, or modify. Using this technique repeatedly, a trail of historical revisions is built that documents how an idea evolved, providing a safety net to always return to previous versions.

Managing different perspectives is further aided by the fact that canvases are moveable on the grid, allowing a user to rearrange the various canvases according to whichever concern they wish to address. That is, they may move “old” ideas aside, cluster canvases by phases of design, group topics, or even juxtapose different perspectives.

4.1.3 Scraps

Scraps address a subset three of design behaviors from Section 2.2 in Chapter 2: drawing different kinds of sketches (design behavior 1), creating low-detail sketches that have only as much precision as needed (design behavior 2), and the tendency to move from generic representations to more refined ones (design behavior 3).

Scraps leverage a key action that users of a whiteboard naturally perform: circumscribing some area of sketched or written content [37]. They typically do this to indicate some object of sorts, or to simply mark an area as important. In Calico, the act of circumscription using the secondary button on the digital stylus has visually the same result: the area that is circumscribed is highlighted with a thin border and grey background. In addition, however, the area becomes an object in Calico that can be further manipulated and has
several preassigned behaviors. In particular, scraps are implicit groups that are movable, stackable, and relatable. I review these properties one by one.

**Implicit groups.** Scraps build upon the approach taken in Translucent Patches [43], which allows users to explicitly declare an area as a group. Anything that is either entirely circumscribed in the first place or otherwise written or drawn in this area afterwards is automatically part of the group. Consider the literal sketch of a car visible in the bottom right of Figure 4.4. It was first drawn on the canvas, then circumscribed by the stylus to become a scrap. The scrap is now a persistent object with a grey background. Any further additions to the car automatically become part of the scrap.

**Movable.** Scraps are movable. Right-clicking and dragging using the digital stylus will move a scrap and its contents to a different location on the board. This seemingly innocuous action in reality represents a significant improvement over the standard whiteboard: content drawn can be rapidly reorganized. It particularly is important that such reorganization takes
place in the language of the user: elements that they have deemed of sufficient importance to promote to being a scrap are the elements that are moved.

*Stackable.* Moving a scrap to a position where some part, or all of it, overlaps another scrap attaches it to the scrap behind it, allowing users to quickly create a stack of scraps (thereby creating hierarchically composed groups), as one would a pile of papers. For instance, the scraps labeled "Speed" and "Position" in Figure 4.4 are part of the scrap labeled "Car Component." If "Car Component" is moved, "Speed" and "Position" are moved as well. Dragging a scrap off of another scrap un-groups it. Moving the scrap labeled "Position" from its current location on the canvas to where it overlaps the scrap containing the intersection will ungroup it from "Car Component" and group it with the intersection scrap in one fluid motion. Note that dragging a scrap implicitly moves it to the top of the order of scraps; scraps do not slide under other scraps.

*Relatable.* By dragging the digital stylus from one scrap to another, an arrow is created between two scraps. The arrow is persistent and anchored to the places where it originated and ended. When scraps are moved, the arrows move accordingly and keep the two scraps related. In Figure 4.4, the scrap "Position" relates to the intersection scrap, and the scrap "Car Component" relates to the car scrap.

Complementing these basic scrap behaviors are several more specialized behaviors accessible via a small radial menu on the scrap itself. First, a scrap’s content can be dropped onto a scrap behind it or, if there is no scrap behind it, back onto the canvas. This allows designers to combine content from multiple scraps. Second, scraps can be given a more regular box shape, and, third, scraps can be made transparent. This latter functionality supports divergence at the micro level of individual scraps (as opposed to the macro level of entire canvases through the grey tabs). By drawing on a translucent scrap that overlays another scrap, developers can explore modifications to this other scrap’s content without overwriting its content. If they are satisfied with the result, they could choose to combine the two scraps by dropping
the content of the translucent scrap. Alternatives can be compared in this manner as well via multiple translucent scraps capturing different deviations.

Scraps are also fundamental to addressing the behavior of using a mix of notations. By virtue of having physical presence, scraps provide a natural basis for serving as a representation for more structured, though still informally drawn, figures, such as user interface sketches or class diagrams. Since scraps are amorphous and take on the shape of a designer’s stroke, shaping them in visually identifiable forms (e.g., boxes, circles, buttons) allows the designer to informally convey a certain meaning. Particularly when combined with arrows, this supports sketching of numerous types of box-and-arrow-like diagrams.

4.1.4 Palette

The palette is the third feature that I incorporated in response to the subset of design behaviors I target from Chapter 2. It specifically addresses the issue of using different types of sketches (design behavior 1).

Palettes are not new, but their typical incarnation in drawing programs is to include a prepopulated set of figures that are not configurable beyond changing the entire set. In design, however, it is not uncommon that a spontaneous notational convention emerges that does not necessarily adhere to any pre-existing or fixed set of figures. Calico’s palette, thus, starts empty, and is filled with content by the designer, enabling the reuse of sketched elements. This allows a temporary vocabulary to be created and leveraged within a design session (as exemplified by the impromptu notations in Figure 4.10, as further described in Section 4.3.2).

The palette leverages scraps for this purpose. Designers can store a scrap simply by dragging it into the palette on the side of Calico’s canvas. The palette has a number of cells, each of
which may hold one or more scraps. By dragging from a palette cell onto the canvas, any
scraps inside that cell are copied to the canvas at the position of the stylus. Once populated,
the user can rapidly create variations of a design simply by dragging key scraps from the
palette. Note that the palette serves as a global clipboard; scraps can be stored and reused
from any canvas.

4.1.5 Implementation Notes

Calico Version One’s implementation consists of approximately 37,000 lines of code written
in Java 6.0. Calico was developed using the Eclipse development environment, and was
tested on Hitachi FX-Duo Starboard interactive whiteboards. It is portable across several
operating systems (i.e., Windows XP/Vista, Linux, and Mac OSX), though all of my labora-
tory evaluations described below were performed using a Starboard connected to a Windows
laptop that ran Calico.

Architecture

Figure 4.5 provides a summary of the architecture of Calico, which is based on the model-
view-controller pattern. The model is responsible for keeping track of all of the sketched
content, including strokes, scraps, arrows, and any relationships that exist among them.
Strokes and scraps are stored as sequences of raw coordinates; arrows are stored as a start
point and end point. Two relationships are supported. First, when scraps fully contain
other content, whether strokes, other scraps, or arrows, a containment relation is kept so
that, when scraps are moved, copied, or deleted, the contained elements are also moved,
copied, or deleted. Second, an anchoring relation is kept if an arrow’s start point or end
point is within a scrap, so that when the scrap is moved, the corresponding point also
moves. All content is managed by an ObjectHandler, which is responsible to provide not
just convenient access, but also accessory methods for creating and restoring from a serialized backup.

Lastly, the grid of canvases exists virtually as a set of two-dimensional coordinates within the components in the model. Each stroke, scrap, or arrow has attached an (x, y) identifier to which canvas it belongs.

The Controller is responsible for interpreting the actions of the user on the canvas and translating those, if needed, to Calico actions. The Gesture Controller is the primary point of access, taking as input the mouse events that are generated when the user interacts with the Hitachi Starboard (I use mouse events only and specifically chose not to take advantage of any special Hitachi Starboard features in order to avoid limiting ourselves to just this electronic whiteboard). Based on the location of the stroke and any pre-existing content that may already exist on its path, the Gesture Controller distinguishes strokes that draw on the canvas from strokes that indicate some action to be performed (see also Section 4.1.5). Strokes that draw on the canvas are passed on directly to the Canvas Controller for creation in the model. Strokes that represent actions are passed on to the Canvas Controller as “action objects” using a command pattern, in order to facilitate future extensions with other gestures and actions.

The User Interface is built using Piccolo [8], a zoomable interface that optimizes screen refreshing. Piccolo was chosen because of its available source code and relatively lightweight footprint. The Shape Painters, UI Widgets, Palette, and Grid all extend the Piccolo Framework and comprise the interface with which the user interacts. The Shape Painters mirror the data model using nodes from the Piccolo Framework to draw content on the canvas; any changes to the data model are then automatically propagated. The UI Widgets take care of Calico functionality such as the tabs, panning, and undo and redo. The Palette also exists as a separate UI widget, albeit with special functionality to hold scraps that can be reused. Upon initialization, Calico instantiates a fixed number of canvases and stores them in the
Figure 4.5: Calico architecture.
grid, which can be referenced using the (x,y) identifier.

**Gesture-Based Input**

Permeating across the subset of four targeted design behaviors from Chapter 2 is the need for gesture-based input. Many of the creative, exploratory activities that Calico supports rely on the fluidity and quickness of sketching. Interactions supplemental to the primary sketching activity should be equally quick. The design flow must be maintained, shapes should be created the way designers want them, and the effort of making changes should be low. Essentially, “viscosity” [63], the cost of making changes, has to be low or designers will be discouraged from exploring opportunities in design.

In order to maintain the natural feel of sketching in the interaction, I have chosen to use a mixed mode approach: many features are ready-at-hand through simple gestures, but some more esoteric functionality that would otherwise require integration of complicated gestures is available in a pie menu off of scraps (visible in Figure 4.6(d)).

Interaction is driven by a small set of context-sensitive gestures, as shown in Figure 4.6. Each stroke of the stylus performs a unique action depending on where it travels. For example, dragging the pen while holding the secondary stylus button on the canvas creates a scrap, but performing the same action on a scrap will move the scrap. With ordinary strokes, dragging the stylus along the canvas draws directly onto the canvas, but slashing it across a scrap, as shown in Figure 4.6(b), will delete it. The gestures take into consideration where the stylus begins, which object(s) it intersects along its path, and where the stylus ends. This is sufficient to build a straightforward interaction scheme that allows the user to create scraps, manipulate them in various ways, draw arrows from one scrap to another, and navigate the entire Calico interface with ease. Each action has a priority, such that stylus strokes spanning multiple interactions behave predictably. For instance, when a user draws a line
from one scrap to another that crosses an existing arrow, a new arrow is created and the existing arrow is not deleted, despite the fact that the stroke struck through it (a gesture that normally deletes that arrow).

### 4.2 Experimental Design

To evaluate Calico Version One, I conducted an exploratory, comparative study in which I asked pairs of participants to perform a software design task using either Calico or a regular whiteboard. My goals were threefold: first, I wanted to ensure that, when Calico is used for an actual design activity, people moved beyond basic whiteboard sketching and used Calico’s advanced features. Second, given that Calico was designed to support specific design behaviors, I wished to observe whether these behaviors actually did occurred. Lastly,
I wished to assess the structure of the design conversation in the pairs that used Calico, and compare it with that of the pairs that used the whiteboard.

### 4.2.1 Recruitment and participants

Sixteen pairs of participants (32 participants in total) were given one hour and fifty minutes to complete a given design task. Eight of the pairs used Calico to perform the design task, and eight of the pairs used the regular whiteboard. I recruited participants at my university using email advertisements and snowball sampling. In a pre-experiment survey, I asked participants to declare their area of expertise, and their industrial experience. All participants were computer science graduate students with some degree of experience designing software systems. The average amount of industrial experience was three years, and ranged from none to seven years of experience. To balance pairs, I paired participants with the similar industrial experience together to avoid having the more experienced designer take over the session. I then gave each pair a rating based on the average experience of its members, and distributed these pairs equally among the whiteboard and Calico conditions so that each condition had an equal amount of experienced and inexperienced pairs.

### 4.2.2 Procedure

Each session lasted between two and two a half hours. When participants arrived, they were given an informed-consent form that included details about the nature of the study. Those in the Calico session were given a 30-minute tutorial, while those in the whiteboard activity were allowed to begin immediately, upon which they were given the two page prompt (see below). Participants were asked to use the whiteboard and to not write on the prompt or any other paper so that cameras could have a clear view of anything they wrote. Pairs were given one hour and fifty minutes of design time, after which they briefly recapped their
design in their own words (5-10 minutes). After pairs finished their design activity, they were given five minutes to collect their thoughts and then to summarize their design in a ten-minute explanation. Afterward, I interviewed them for ten minutes about their opinions and experience concerning the activity, and their past experience with similar technology. At the end of the session, each participant was given $100 as an incentive and a $250 prize was awarded to the pair with the best design. As a result, all pairs took the exercise seriously and were fully engaged throughout the design exercise.

Each session was recorded on video camera for subsequent analysis. Additionally, Calico produced detailed logs that captured each individual user action (e.g., scribble drawn; scrap created, moved, or deleted; switch canvas in the grid).

### 4.2.3 Task

Each pair received the same design prompt, asking for the design of an educational traffic signal simulator to be used by students in a civil engineering course (the same prompt was used in the Studying Professional Software Design workshop; it is included in its entirety in the introduction to the Design Studies journal special issue dedicated to this workshop [65]). The prompt, provided a series of open-ended goals and requirements, asking the pairs to design a system that allowed engineering students to: (1) create a visual map of the roads, (2) describe the behavior of the lights at each intersection, (3) simulate traffic flow, and (4) change parameters of the simulation, such as traffic density. The prompt also instructed pairs to produce a design that they could present “to a pair of software developers who will be tasked with implementing it.”
4.2.4 Measures

Our goal was to: (1) measure how often Calico’s advanced features were used, (2) examine how those features were used with respect to the design behaviors I outlined earlier, and (3) assess the structure of the design conversation in both the whiteboard and Calico sessions. I also asked a standard set of exit interview questions concerning the group’s satisfaction with the tool and the design process that they followed.

Use of features. In order to objectively verify that participants move beyond just sketching like they would on a regular whiteboard and actually use the advanced features of Calico, I measured the amount of usage that each feature received. To perform this, I reviewed the videos and made a note of each time a particular feature was used.

Design behaviors. After I had recorded all instances of the features being used, I performed a qualitative analysis of how the features were used. I was interested in seeing how well those features supported the targeted subset of four design behaviors from Chapter 2. For the first design behavior, using different types of diagrams, I watched out for situations where participants mixed several notations together in a single canvas, and highlighted occurrences of impromptu notations that were unique to a particular pair and did not appear in others. For the second design behavior, drawing only what they need, I noted the general diagrams that participants sketched, and paid particular attention to what models participants created, and if and how they used scraps to do so. For the fourth design behavior of moving from abstract to concrete, I compared photos of diagrams at various stages, and noted how pairs restructured their diagrams and added additional detail. When considering the fifth design behavior, navigating between perspectives, I was particularly attentive to the reason that participants moved out of one canvas and into another within the grid, and I classified the general activities that occurred in each of the canvases as well.

Structure of Design Conversations. After qualitatively assessing the videos for evidence of
the design behaviors, I assessed the impact that Calico had on the structure of the design conversation by coding audio transcripts of the design sessions for what kinds of activity took place at each moment. The categories and guidelines for the coding scheme were adopted, with minor changes, from a previous study on design meetings [56]. As the authors of that study explain, the categories were derived from the Design Rationale literature [49] as well as studies of group activity [67], and reflect key aspects of the design activity. With respect to the activities from design rationale, statements were separated into Issues at hand, the Alternatives or solutions raised, and the Criteria used to evaluate an idea. Within the same coding scheme, statements could also be organized into organizational activities, i.e., conversations the group had to organize itself (specifically called Meeting Management, Summary, Walkthrough, and Goal), to Clarify their ideas, or to engage in Digressions. Two additional categories not part of the original coding scheme, Technology Management and Technology Confusion, were inspired by a previous study based on the same coding method that analyzed a tool’s effect on the design process [58]. Technology Management refers to the times when the participants’ focus was devoted to the tool itself rather than the activity at hand, and Technology Confusion refers to time lost due to system failure. Lastly, any category that did not fit into any of the above was categorized as Other.

Due to limited time and resources, only a subset of the sessions, specifically six Calico and six whiteboard sessions, were coded (see Table 4.3 and Table 4.4). While this is not enough to claim statistical significance, it is enough to gain a sense if there is, in fact, an apparent difference in process between the two conditions.

In order to verify the validity of the coding, I performed an interrater reliability test, and also consulted with researchers who previously applied this coding in their own past studies. After initial training, two individuals independently coded a session, and then compared their coded transcripts to perform an interrater reliability test. I obtained a Cohen’s k value of 82% at this level of granularity. I then compared the total time that each coder had for
each category, and found a correlation of .998. These measures are well within the accepted
tolerance for behavioral analysis.

*Satisfaction and perceptions of participants.* At the end of each session, I interviewed partic-
ipants in order to learn how Calico affected their approach to the design task, and what the
positive or negative aspects of Calico were for the users. I did this by first asking participants
to reflect on their designs and the process they used to get there. I then asked them explain
how they used each of the advanced features, their overall satisfaction with each feature, and
whether they had any suggestions for improvement.

### 4.3 Results

The discussion of my results is organized by the major categories of analysis described in the
previous section: use of features, design behaviors, design conversations, and satisfaction. In
this section, I only present the results that I observed; I do not make attempts to interpret
them. In Section 4.3.2, I bring the results together and draw my conclusions about the value
of Calico.

#### 4.3.1 Feature use

I first focused on whether participants would move beyond just basic sketching to using the
advanced features of Calico. This point is important, given that there was no incentive for
participants to use any of the advanced features other than the features being useful to their
task at hand; they could have instead chosen to sketch as they normally would on a standard
whiteboard without the help of scraps or the grid.

Figure 4.7 shows the result, marking each time the grid (a), scraps (b), or palette (c) was
Figure 4.7: Feature usage across the eight calico pairs.
used. From the graph, it can be seen that the grid was unanimously used by all pairs, with heavy usage by most pairs and moderate by some. The pairs predominantly switched to the grid view first for navigating to different canvases, though as time went on, several incorporated the use of the tabs to navigate to adjacent canvases (see Section 4.3.2).

There was a wide distribution in the frequency of scrap use, with some pairs strongly relying on scraps, others exhibiting more moderate use, and a few barely using them. The palette was the least used feature, with just two pairs using it some and three other pairs using it twice each.

4.3.2 Design behaviors

Given my goal of supporting the specific subset of four design behaviors of the total set in Chapter 2, this section focuses on occurrences of these behaviors in the various pairs, as well as how the advanced functionality of Calico was used in the presence of these behaviors.

Design behavior 1: Drawing different types of diagrams

As expected, numerous notations were used by the participants as part of their design process. These include class diagrams, user interface components, and diagrams specific to the domain of traffic simulation. These different types of diagrams were often created in response to one another. In the whiteboard sessions, this led to heterogenous content in different notations spread out over the whiteboard. In the Calico sessions, the pairs broke up their designs across different canvases, as exemplified in Figure 4.12, and as a result most canvases used just a single notation. Pairs sometimes did mix different notations in a single canvas, such as in the Figure 4.9(d), where one of the pairs uses representations of a traffic intersection in the top of the image with pieces of code representations in the bottom-right of the image,
on the same canvas. This pair, as well as other pairs that similarly mixed representations, used the different representations to juxtapose different views of the design.

Pairs had no ready notation to represent traffic structures, and so they created their own on the fly. For instance, the image in Figure 4.8(a) shows how one pair used scraps to model a state diagram that is part of their vision for how civic engineering students will specify the timing of traffic lights in their simulation. Note how the state diagram exists on one of two tabs meant to be part of the user interface. Next, in Figure 4.8(b), the same pair applied a similar notation to define the logic that is executed once a car arrives at an intersection. Here the pair reused the same diagrammatic elements by copying scraps. In Figure 4.8(c), another pair created dozens of small scraps and placed them in a grid configuration to simulate the interface for traffic flow. They filled in select canvases with the pen to simulate a particular route, and experimented with different routes by erasing and filling in other scraps. In all three examples within Figure 4.8, pairs were able to create copies of their scraps and reuse them to attempt alternate combinations.
Figure 4.9: Examples of models in varying amounts of detail.

**Design behavior 2: Drawing only what they need**

I observed the use of low-detail models in every single design session. The pairs that used Calico created the same type of low-detail models that I saw on the whiteboard, and in most cases they used scraps to create these models. Of the eight Calico pairs, two simply sketched directly on the canvas as they would have on a traditional whiteboard, and did not further manipulate their sketches. In the cases where pairs did use scraps, they benefited from the extra functionality, such as moving scraps to reorganize a design, copying them in order to create variations, and arrows to define relationships.
The images in Figure 4.9(a) and 4.9(b) illustrate an example of how many of the pairs used scraps to create box-and-arrow representations of code structures. The pairs that did not use scraps, such as the pair that produced Figure 4.9(c), drew boxes and arrows, but did not heavily restructure their diagrams. The pairs that did use scraps engaged in more organization of their box-and-arrow diagrams through the moving and copying of scraps. Additionally, they typically refined their scrap-based models into UML-like models, showing how scraps can be used as the basis for such refinement from low detail to more detail. In the case of Figure 4.9(b), the designers annotated some of their scraps with “I” to indicate that it is an interface, and on the right-hand side they created lists of parameters for each object.

Many of the other diagrams produced in Calico shared similar qualities with the low-detail diagrams, with Figures 4.9(c) and 4.9(d) providing two more examples. These diagrams were quick sketches that participants initially used to help them understand a particular situation. The designers in Figure 4.9(c) did not use scraps heavily and so they experimented with traffic configurations by simply drawing and erasing different scenarios. The designers in Figure 4.9(d) used scraps to accomplish a similar task. In this case, they used scraps of different shades to represent traffic configurations, and experimented with different configurations by moving and copying the scraps. Additionally, they used scraps to pull pieces of the design from surrounding canvases in order to understand how a particular traffic configuration would work within the context of other parts of the system.

**Design behavior 3: Refinement of representations**

Both the grid and scraps were used in the refinement of representations over time. At the most general level, the grid partitioned the design space so that participants could separate high level and low level representations. Many pairs used spatial orientation to navigate from higher level representations to lower level representations, where higher level
representations would exist at the typically left-most canvases, and the details of components in adjacent canvases to the right or below the originating canvas. Next, when members made the transition to more concrete representations, they would transform sketches into scraps, and create copies on other canvases where they would expand them in more detail and create relationships between them.

There were two patterns of this behavior that occurred frequently within pairs. Figure 4.10 illustrates a representative example of the first pattern, which occurred in over half of the Calico pairs. Most pairs, but not all, began by creating several lists of design requirements, goals, and what they viewed as major aspects of the system. The example in Figure 4.10(a) contains several high-level components such as Maps, Intersections, Cars, and so on. After brainstorming lists such as these, pairs would commonly convert them into scraps (Figure 4.10(b)) and copy them into another canvas, where they would expand on these in more detail, as in Figure 4.10(c). However, creating lists such as these was not common in all pairs, two pairs chose to jump right to diagramming. In the exit interviews, these two pairs both reported that they were first concerned with understanding the requirements and the world that they were modeling. They both began by creating concrete representations of what they knew was true within intersections, and then developed a high level understanding by drawing on top of and discussing these models with their partner. To accomplish this, they created many diagrams of intersections, and from these jumped to writing down assumptions in lists contained in other canvases. When they encountered a new concept that they sensed was complex, such as how to handle the timing within intersections, they would loudly say, “let’s leave this for later”, and leave it as a generic annotation.

A second pattern that occurred more frequently was the breaking down of the design across multiple canvases. A representative example is depicted in Figure 4.11, in which the participants from that pair partitioned their code structure across many spaces. They created a high level perspective of the architecture in one canvas using scraps (Figure 4.11(a)) and
lists converted to scraps. In Figure 4.11, the participants divided the architecture of their program into UI, Map, and Simulation Logic. They copied the contents of this canvas to an adjacent canvas, where they subsequently fleshed out the details of Simulation Logic, but left Map and UI with no additional detail. They then used a third canvas to work out the details of Map, while leaving Simulation Logic and UI as is. Using this divide and conquer strategy, the participants were able to effectively partition their design. This behavior of spreading diagrams across several canvases is similar to what Dekel and Herbsleb [22] observed in their studies, where diagrams in one drawing space would depend on references located in other spaces.

**Design behavior 5: Navigating between perspectives**

As the heavy use of the grid would suggest, Calico assisted designers readily in shifting focus. All sessions resulted in grids similar to the one shown in Figure 4.3, with the designers moving back and forth among canvases frequently. Pairs switched to a new canvas to either: (1)
(a) High level perspective of system architecture  
(b) Definition of Simulation Logic  
(c) Definition of Map

Figure 4.11: Representations of software components were broken up across several canvases.

(a) List of requirements in bullet point form  
(b) UI mockup of the traffic simulator interface  
(c) Architecture in the form of boxes and arrows

Figure 4.12: Participants tended to focus their efforts towards a particular theme in each canvas.
address additional detail generated by the current sketch, or (2) generate a new alternative. In the majority of cases it did not matter where on the grid a pair went to continue their design activities, as a consequence of which I observed quite a few linear chains of canvases on the grid in use. Several interesting cases did emerge, however. For example, in one session, each of the pair members “owned” a grid row of design content in that they were the primary designer for all of the sketches in their row. As another example, one pair used the spatial metaphor extensively, with members talking about moving in a given direction (i.e., left, right, up, down) to navigate to certain sketches. While not all pairs operated as explicitly in terms of direction, the grid’s spatial orientation did provide a consistent layout that enabled the pairs to shift focus by navigating to different canvases that they knew existed in certain locations.

A number of pairs exhibited bursts of back-and-forth switching between canvases to compare content, as illustrated in Figure 4.7(a) by the overlapping sequences of dots. This happened for two reasons. First, participants moved back-and-forth when they needed to mentally juxtapose two concepts, such as when they were working to improve their design for the user interface in conjunction with the underlying model. Second, they would take stock to verify that the different sketches were consistent with one another to make sure they had not inadvertently introduced problems when they furthered some aspect of their design.

The contents of the canvases tended to break down in three categories: (1) requirements, (2) code structures, and (3) UI mockups, with examples of each depicted in Figure 4.12. Every pair used the first canvases to lay out their requirements in a list, as in Figure 4.12(a), after which they would move to another canvas to embark on the design proper, producing diagrams similar to Figures 4.12(b) and 4.12(c). Pairs would frequently return to the requirements for inspection and assessment of their progress.

The ability to copy canvases proved useful not only to generate alternatives (one can see several variant sketches in Figure 4.3), but also when it was desirable to divide the contents
of a canvas between two canvases so to be able to work on each part separately. Copy followed by subsequent deletion of the respective halves of the copied content achieved this desired result. One of the pairs used the copy feature to create backups of individual canvases, and moved these backup copies out of the way by dragging the canvases to the outer edges of the grid. Several other pairs moved canvases in the grid around to more clearly organize their design.

Overall, I observed that participants use the grid as a tool to divide design content across canvases, where a shift in attention was commonly accompanied by a change of canvas, and long periods of attention were marked by extended periods of continued activity in a single canvas (visible as gaps between points in Figure 4.7). Additionally, I observed the spatial layout to be helpful to designers, both in providing clear references to design content and in organizing the overall design effort.

4.3.3 Structure of design conversations

I now examine the structure of the design conversations within the Calico pairs, and see how it compares with the pairs that performed the activity on the whiteboard.

How time was spent

I first determined the structure of the design process by measuring the total time spent in each design category during conversation. I coded the transcripts of the spoken parts of the meetings, summarizing the total time spent in each category, as shown in Table 4.2.

Categories related to design rationale (i.e., issues, alternatives, and criteria) took up the majority of the design sessions. In the Calico pairs, these took up 67.5% of the meeting time, and in the whiteboard pairs they took up 62.7% of the meeting. Both Calico and
Table 4.2: Summary of design conversation categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Whiteboard</th>
<th></th>
<th>Calico</th>
<th></th>
<th>Sum</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pairs</td>
<td></td>
<td>Pairs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>6.37</td>
<td></td>
<td>7.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Issue</td>
<td>1.06</td>
<td></td>
<td>0.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alternative</td>
<td>37.04</td>
<td></td>
<td>41.83</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Alt</td>
<td>2.08</td>
<td></td>
<td>1.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>15.63</td>
<td></td>
<td>16.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Criteria</td>
<td>0.48</td>
<td></td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>62.66</td>
<td></td>
<td>67.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General Clar</td>
<td>0.04</td>
<td></td>
<td>0.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artifact Clar</td>
<td>0.11</td>
<td></td>
<td>0.21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>1.32</td>
<td></td>
<td>1.57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Summary</td>
<td>0.00</td>
<td></td>
<td>0.41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walkthrough</td>
<td>2.05</td>
<td></td>
<td>1.90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Walk</td>
<td>0.00</td>
<td></td>
<td>0.07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal</td>
<td>1.07</td>
<td></td>
<td>0.91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Goal</td>
<td>0.52</td>
<td></td>
<td>0.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meeting Mgmt</td>
<td>6.37</td>
<td></td>
<td>7.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Meeting Mgmt</td>
<td>0.00</td>
<td></td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digression</td>
<td>0.82</td>
<td></td>
<td>0.48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clar Digression</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Technology</td>
<td>0.35</td>
<td></td>
<td>3.66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology Conf</td>
<td>0.04</td>
<td></td>
<td>5.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>0.00</td>
<td></td>
<td>0.91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.43</td>
<td></td>
<td>22.79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pause</td>
<td>24.01</td>
<td>24.01</td>
<td>9.67</td>
<td>.67</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

63
the whiteboard had a similar breakdown within the design rationale categories. Within the Calico and whiteboard pairs, discussing alternatives occupied 41.8% and 37.0% of the time, respectively. Both pairs spent roughly the same amount of time discussing issues, at 7.3% and 6.4% for Calico and whiteboard pairs, respectively, and also the same amount of time evaluating solutions in the criteria category, with 16.5% and 15.6% for Calico and whiteboard, respectively. However, the average length of stay in each category, i.e. the average uninterrupted time for a particular category, was higher for the Calico pairs than the whiteboard pairs. The average length of stay for alternative in Calico pairs was 13.0 seconds compared to 8.7 seconds for whiteboards. The average length of stay for criteria in Calico pairs, 11.0 seconds, was also longer than that for the whiteboard pairs, 7.5 seconds. However, the average length of time per issue proposed was similar, with Calico pairs averaging 8.1 seconds and whiteboard pairs average 7.1 seconds.

Roughly double the amount of time was devoted to management related activities in Calico pairs compared to whiteboard pairs, however this time difference was due to discussion of the technology. Management related categories occupied 22.8% of the meeting in Calico, while they only occupied 12.4% of the meeting in the whiteboard pairs. Of that time in Calico, 8.8% was spent discussing the technology (combined sum of Management of Technology and Technology Confusion categories). Both pairs spent relatively the same amount of time explicitly discussing meeting management, 7.3% and 6.4% for Calico and whiteboard pairs, respectively. The other categories were roughly similar as well.

I also recorded the amount of time spent not talking, which I recorded as pause. I found that the whiteboard pairs were quiet for more than double the period of Calico pairs, with whiteboard pairs silent for 24.0% compared to 9.7% for Calico pairs. Upon closer inspection, I saw that the majority of whiteboard pairs used this period of silence to work independently of each other. Calico pairs were not given this opportunity since the system only permits one user to write at a time. While I do not measure the impact that this had here, during the
exit interview participants reported frustration over their inability to work independently. As a result, they reported that they had to force their attention on what the other person was writing, and abandon ideas they were thinking of independently because they could not write them down.

I then recorded time spent clarifying answers across all categories, though I found that little time was dedicated to clarification within both sessions. The combined percentage of clarification across all categories for the Calico pairs was 2.8%, while it was 4.3% for the whiteboard pairs.

Lastly, I examined pairwise levels of similarity across all pairs, shown in Table 4.3. I compared the average time spent within each category across Calico and the whiteboard conditions, and found a correlation of .92, indicating a strong relationship. All Calico pairs had a strong correlation with one another, ranging from .90 to .98. Within the whiteboard pairs, there was a much larger variability, with correlations ranging from .16 to .98, with a median of .58. A major way in which the meetings varied, particularly between Whiteboard Session 2 and Session 5, was in the amount of time that was categorized as pause. Removing the pause category from the pairs raises the correlations significantly to the range of .93 to .99 (not visible in table). Overall there appears to be much similarly across all sessions, with pairs within the Calico condition having the tightest range when considering all of the categories.

**Transitions Between Activities**

In addition to total time, I also examined all incoming and outgoing transitions that pairs made between categories in order to understand the flow of design conversations. I first computed the total number of times the participants transitioned from one category to another, which is shown in Figure 4.13 through arrows, the thickness of which representing the relative frequency that a particular transition occurred. The most common transitions
Table 4.3: Correlations between sessions for total time spent (dark cells pertain only to Calico sessions, white cells pertain only to whiteboard sessions, and lightly shaded cells are the intersection of both).

<table>
<thead>
<tr>
<th></th>
<th>C.1</th>
<th>C.2</th>
<th>C.3</th>
<th>C.4</th>
<th>C.5</th>
<th>C.6</th>
<th>W.1</th>
<th>W.2</th>
<th>W.3</th>
<th>W.4</th>
<th>W.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.2</td>
<td>0.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.3</td>
<td>0.98</td>
<td>0.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.4</td>
<td>0.97</td>
<td>0.96</td>
<td>0.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.5</td>
<td>0.94</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.6</td>
<td>0.95</td>
<td>0.95</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.1</td>
<td>0.94</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
<td>0.99</td>
<td>0.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.2</td>
<td>0.41</td>
<td>0.20</td>
<td>0.44</td>
<td>0.33</td>
<td>0.27</td>
<td>0.43</td>
<td>0.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.3</td>
<td>0.92</td>
<td>0.97</td>
<td>0.94</td>
<td>0.95</td>
<td>0.98</td>
<td>0.94</td>
<td>0.98</td>
<td>0.20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.4</td>
<td>0.92</td>
<td>0.98</td>
<td>0.94</td>
<td>0.98</td>
<td>0.99</td>
<td>0.95</td>
<td>0.98</td>
<td>0.16</td>
<td>0.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.5</td>
<td>0.61</td>
<td>0.55</td>
<td>0.73</td>
<td>0.67</td>
<td>0.61</td>
<td>0.74</td>
<td>0.61</td>
<td>0.92</td>
<td>0.54</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>W.6</td>
<td>0.91</td>
<td>0.91</td>
<td>0.95</td>
<td>0.93</td>
<td>0.93</td>
<td>0.97</td>
<td>0.93</td>
<td>0.57</td>
<td>0.91</td>
<td>0.90</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Figure 4.13: How time was spent within and between the design activities.
that occurred belonged to the design rationale categories: *issues*, *alternatives*, and their *criteria*. In Calico they were involved in 76.8% of all transitions, and 77.7% of all transitions in the whiteboard pairs.

I then computed the correlation matrix for the Calico and whiteboard pairs, and found striking similarities between the transitions of all 22 categories. In order to get the correlation matrix, I computed the average percentage that a given transition happens between one category to the next, and established the correlation matrix shown in Table 4.4. The average transition that happens in the Calico pairs was strongly correlated with that of the whiteboard pairs at .92. As with the previous table of correlations, the Calico pairs, on average, are more highly correlated with one another than the whiteboard pairs and have a tighter range. The Calico pairs range from .63 to .93, compared to whiteboard pairs, which range from .41 to .98. As with the previous table of total time spent, whiteboard pairs 2 and 5 also have low correlations with the other whiteboard pairs, and a high correlation with each other, suggesting a distinctive difference in work style. Also, unlike the previous table, Calico pair 4 correlates slightly less, on average, than other Calico pairs, and much less with whiteboard pairs, suggesting they used a slightly different process as well.

Additionally, I looked for patterns of transitions. In order to discover patterns, I use the same method described in Olson et al. [57]. In this method, I observe for transitions that occur more often, or less often, than they are statistically expected to. In order to determine the expected value, I first calculate the number of times that a particular category occurs, and divide that number by the total number of instances of all categories, which gives us the expected percentage of occurrence. I then compare this expected percentage with the actual percentage for a particular transition. If no structure exists in the design session, then the probability of moving from one activity to the next will be the same as the expected percentage. However, if a pattern exists, then I will observe a deviation between those two numbers.
Table 4.4: Correlations between sessions for transitions (dark cells pertain only to calico sessions, white cells pertain only to whiteboard sessions, and lightly shaded cells are the intersection of both).

<table>
<thead>
<tr>
<th></th>
<th>C.1</th>
<th>C.2</th>
<th>C.3</th>
<th>C.4</th>
<th>C.5</th>
<th>C.6</th>
<th>W.1</th>
<th>W.2</th>
<th>W.3</th>
<th>W.4</th>
<th>W.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.2</td>
<td></td>
<td>0.82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.3</td>
<td>0.86</td>
<td>0.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.4</td>
<td>0.70</td>
<td>0.75</td>
<td>0.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.5</td>
<td>0.90</td>
<td>0.88</td>
<td>0.89</td>
<td>0.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.6</td>
<td>0.90</td>
<td>0.86</td>
<td>0.95</td>
<td>0.76</td>
<td>0.93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.1</td>
<td>0.83</td>
<td>0.83</td>
<td>0.83</td>
<td>0.45</td>
<td>0.94</td>
<td>0.84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.2</td>
<td>0.45</td>
<td>0.45</td>
<td>0.66</td>
<td>0.45</td>
<td>0.46</td>
<td>0.64</td>
<td>0.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.3</td>
<td>0.85</td>
<td>0.84</td>
<td>0.80</td>
<td>0.46</td>
<td>0.93</td>
<td>0.83</td>
<td>0.98</td>
<td>0.41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.4</td>
<td>0.88</td>
<td>0.88</td>
<td>0.82</td>
<td>0.59</td>
<td>0.90</td>
<td>0.87</td>
<td>0.91</td>
<td>0.47</td>
<td>0.93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W.5</td>
<td>0.58</td>
<td>0.66</td>
<td>0.81</td>
<td>0.49</td>
<td>0.65</td>
<td>0.76</td>
<td>0.62</td>
<td>0.88</td>
<td>0.57</td>
<td>0.62</td>
<td></td>
</tr>
<tr>
<td>W.6</td>
<td>0.88</td>
<td>0.88</td>
<td>0.80</td>
<td>0.54</td>
<td>0.91</td>
<td>0.86</td>
<td>0.94</td>
<td>0.42</td>
<td>0.95</td>
<td>0.97</td>
<td>0.60</td>
</tr>
</tbody>
</table>

Figure 4.14: Emergent pattern of transitions across both the Calico and whiteboard pairs.
Using the above method, I observed sequences of up to four transitions, and a pattern began to emerge that was present in both the Calico and whiteboard pairs. The pattern that emerged is shown in Figure 4.14. The solid lines in Figure 4.14 represent strong trends, while the dashed lines represent transitions that are weaker, but still occur well above probabilistic levels. M stands for any management category, I for issue, A for alternative, and C for criteria, in accordance with the categories in Figure 4.13. There was a strong tendency to make frequent transitions between alternative-criteria, or between management-alternative. Additionally pairs commonly engaged in serial evaluation, where one participant would challenge the evaluation of the other, leading to new criteria with which to judge the alternative. Within the pure design categories, sequences of alternative-criteria transitions were typically punctuated by either the discussion of an issue, or a category from management. Olson et al. called these combinations IAC or MAC, based on their category names. These specific combinations are examples of “design episodes”, which were found by them to be common in design meetings. I found these design episodes to be present in both the Calico and whiteboard pairs. The episodes of IAC, which has an expected value of .04%, appeared to happen more commonly in Calico pairs, occurring an average of 3.25% of all transitions, compared to 2.33% of all transitions for whiteboard. Episodes of MAC, also with an expected value of .04%, appeared to happen more commonly in whiteboard pairs at 2.64% of all transitions, versus 1.61% of all Calico transitions.

4.3.4 Satisfaction and perceptions of participants

At the end of each session I verbally interviewed the participants and asked them to reflect on their own design process and satisfaction with the tool. Despite some technical difficulties, six out of eight pairs reported that they greatly enjoyed using Calico. The grid was unanimously praised as a tool to organize the design space. One participant praised that “it was the first time [they] felt like they were using a whiteboard [in a digital medium]”, and expressed
that other pen-based software tools that paginated their drawing space, such as Microsoft OneNote, made navigating designs frustrating. Five of the eight pairs explicitly praised scraps as tools for quickly copying content and moving it across canvases, however many expressed frustration that they were not able to rotate or resize them. Two pairs reported mixed feelings about the overall tool, stating that Calico merely felt like a digital whiteboard and did not compensate for the bulky hardware. All pairs reported dissatisfaction with the hardware in general, stating that it made their writing very sloppy, and that single user input was a “deal-breaker.” When asked if they would use Calico in their own design sessions, seven pairs expressed a great deal of interest, but only if it allowed more than one user to draw at a time. All participants reported frustration over this issue, with one participant stating that it caused them to “spin their wheels” while they waited for their partner to finish writing.

Seven out of eight pairs reported that Calico did not alter or interfere with how they would normally approach a design problem, and that the ability to easily copy canvases encouraged them to explore more solutions than they normally would have. Also, pairs reported that the presence of scraps changed the way they approached creating lists, stating that they kept in mind the ability to easily copy list items and transform them into software architecture components. One pair of the eight felt that the benefits of Calico did not outweigh the hardware drawbacks, saying that they could have achieved the same design using a stack of papers. In general, nearly all participants were content with the final design that they had created, with most participants calling their final design “a good start” given the limited amount of time they had to create their design.
4.4 Discussion

Building upon the results from Section 4.3, I now shift to interpreting what the results mean with respect to Calico use in support of software design at the whiteboard.

4.4.1 Calico features

Bringing the results described in Section 4.3 together, it is clear that Calico does provide support for subset of four design behaviors that I targeted from the total set of design behaviors of Chapter 2 and that designers are able to use the features in support of these design behaviors when they so desire. However, it is also clear that features were not always used when they could have supported a certain design behavior and that not every pair utilized every feature.

I interpret the differences in use between the different pairs as stemming from two factors. First, some pairs had early success in using scraps, and discovered ways in which they made their life easier. They subsequently stayed with scraps more persistently. Other pairs simply drew on the canvas and ventured rarely into attempting to use scraps. With some early troublesome interactions (e.g., an accidental deletion by a slash-through, using the wrong stylus button to create a scrap), they became discouraged and used traditional methods instead.

Second, the benefits of using scraps are not always apparent until later. One has to anticipate needing to alter a drawing by creating it using scraps in the first place, otherwise one ends up drawing it twice: once as background sketches and once to create reusable elements from parts of those sketches. This second step represents additional work, and may be skipped in favor of simply redrawing a sketch, especially if it is small or not-too-involved.
In both cases, I believe more training and the build-up of experience over time has the potential to overcome the issue. Simply sketching on a whiteboard is so ingrained, it takes time to internalize and adjust to a new form of interaction with sketched content.

Post-experiment conversations with the participants confirmed some of these observations. While I generally received very positive feedback, several issues were identified. First was that the nature of scraps was not as intuitive to learn as I had hoped. The participants did not think this was an outright failure, but did note the need for more training and more examples of scrap use. The participants also noted that tabs did not inform the user as to whether a neighboring canvas is occupied. Spatial navigation directly from canvas to canvas without first moving to the grid, therefore, is hampered. A simple mark to indicate a neighboring canvas has content, together with a popup on hover, may overcome this problem.

The participants also wished for extra functionality on scraps. Some of the requested functionality is generic in nature and straightforward to add, such as rotation, resize, and different types of arrows. Some, however, is more specific and pertains to the fact that designers mentally assign meaning to scraps and expect functionality commensurate to that meaning. Such expectations arose later in the sessions, when the design had been largely worked out and the designers now wished to provide additional detail to complete the design. For instance, they asked for functionality to refine scraps into UML classes with a name, variables, and functions, or to turn scraps into lists for easy reorganization and requirement tracking. I believe such refinement can be integrated without disturbing scraps’ present sketchy, informal nature, which is so crucial in the early stages of design. Particularly, I envision typing of scraps, with the assignment of type (e.g., UML class, UI element, architectural component, ER element) signaling additional functionality becoming available.
4.4.2 Design behaviors

With respect to how pairs approached the design task, the Calico pairs exhibited some very similar behaviors, while the whiteboard pairs noticeably differed at times. For example, not all whiteboard pairs used transient diagrams, some maintained all of their diagrams for the entire session and refined these over a period of time. In contrast, other whiteboard pairs continuously erased what they drew and often worked from memory. Some pairs chose to work together the entire time, while other pairs worked on their design silently in parallel at extreme ends of the board. One of the pairs that did this eventually did review each others’ design and reconciled their different approaches, while another pair largely operated independently the entire session with few design decisions being contested or brought together. The Calico pairs, however, seemed to be implicitly guided into certain behaviors by the tool, and these behaviors were consistent across the majority of Calico pairs. The abundance of space led pairs to save and evolve their diagrams over time to a much greater extent. Scraps made it possible to reuse, as well as refine, diagrams, which nearly all pairs did. Additionally, the grid caused the pairs to create clear divisions between different aspects of their design. Overall, it appears that Calico’s features led pairs towards the specific behaviors of design that I planned for, while the whiteboard pairs, who did not have these features, varied much more greatly in their approaches. Whether or not these behaviors lead to a positive impact on the overall design is yet to be determined, but, regardless, my observations point to the conclusion that these behaviors are supported by Calico.

4.4.3 Interference

Another important result of the study is that Calico did not interfere with the design activity. If anything, it led to more issues, alternatives, or criterias that were discussed. In the exit
interviews, all designers agreed that Calico did not much effect the nature of their design conversations, and the results from the analysis of the design structure tend to support that. They dedicated relatively the same amount of time of each session discussing design activities at 62.6% and 67.5% for whiteboard and Calico conditions, respectively. Further, the use of Calico had little effect on the amount of time spent managing the activity, at 6.4% and 7.3% for the whiteboard and Calico pairs, respectively. The pairs intuitively understood how to manage their design using the grid, and so they did not need to exert additional effort in managing it. This behavior falls in line with previous research that supports the notion that designers make heavy use of spatial properties to orient themselves and organize their design [55, 11]. In the normal whiteboard sessions, it was not uncommon for participants to look back and forth between two diagrams on opposite ends of the board, and sometimes place a hand on one diagram while they looked at another. Participants used Calico in much the same way by rapidly moving back and forth between two canvases when considering two diagrams. In one exit interview, one pair requested the ability to view two canvases simultaneously in a juxtaposed view to allow a similar activity.

Calico did, however, seem to cause some changes. The Calico tool seemed to force a greater degree of homogeneity between the pairs. The pairs that used Calico had a higher amount of conformity between them with respect to the total time spent in each category than the whiteboard pairs. The correlations between pairs for transitions between categories demonstrated the same tendency. This discrepancy in design conversations between the Calico and whiteboard pairs seems to, by and large, be related to the amount of time spent working independently. Two whiteboard pairs, 2 and 5, spent a significant portion of the session working quietly in parallel, which is reflected in their low correlation with other pairs for time spent in Table 4.3. Interestingly, the transition frequency correlation matrix in Table 4.4 for these two pairs shows the same pattern of discrepancy. This suggests two things: (1) the pairs that chose to spend a significant amount of time working silently in parallel had a different design conversation structure than those whiteboard pairs that did not spend
much time working silently, and, conversely, (2) the pairs that did not spend significant
time working independently had a sequential process that correlated relatively highly with
the Calico pairs. This second point is interesting because it shows that, while many of the
Calico pairs considered themselves handicapped by the inability to work independently, the
structure of their design process was similar to the highly collaborative whiteboard pairs.

4.4.4 Focus

Finally, after comparing the videos with the coded transcripts, it seemed that the pairs that
used Calico had a greater amount of focus on their partner’s ideas and a slightly better
shared understanding of recorded ideas than the whiteboard pairs. While both the Calico
and whiteboard pairs spent a relatively similar amount of total time per session discussing
issues, alternatives, and criteria, the Calico pairs had a larger average of consecutive time per
alternative and criteria discussed. The average length of continuous time that an alternative
was 50% longer in Calico pairs than whiteboard pairs (13 seconds versus 8.7 seconds) and
the average length of continuous time a criteria was expressed was also nearly 50% longer
in Calico pairs than whiteboard pairs (11 seconds versus 7.5 seconds). While this was, in
part, due to the single input interface of the electronic whiteboard, other factors in Calico
forced the focus of participants well. Within Calico, the pairs would partition their design
into different canvases in the grid, and so the drawing spaces in Calico tended to have a
tighter focus than the whiteboard, which shows all of the content at once. As a result,
the participants in the Calico pairs were not distracted by drawings within other parts of
the design, whereas the whiteboard pairs could gaze at any part of their design at any
time. When a participant in a Calico pair did refer back to other parts of the design, they
had to switch to an entirely different canvas via the grid, effectively forcing their partner’s
attention to move with them to that part of the design. While Calico pairs were not able to
move between topics as quickly as whiteboard pairs, it led to more elaborate explanations
and longer individual responses when presenting a criteria. Within the whiteboard pairs, it would sometimes be the case that while one partner would be explaining their solution, the other would ignore them while they considered another part of the design. The split attention was reflected in the longer time spent clarifying ideas for the whiteboard pairs, which was 53% longer than Calico pairs (4.3% of the total session for whiteboard pairs, on average, versus 2.8% of the total session for Calico pairs, on average).

4.4.5 In sum

Overall, my detailed analysis of how Calico was used and influenced the design behaviors and conversations, as well as the positive responses to Calico from the participants, show that Calico has promise in enhancing design at the whiteboard. More elaborate training is likely needed for users to properly take advantage of scraps, but I still saw that Calico’s features are helpful, while avoiding impeding the design process.

Calico’s strength lies in its support for the informal process of design, during which the emphasis is placed more on unstructured exploration and less on a precise analysis of the design. Some pairs benefited more than others from Calico’s features, but overall, Calico assisted them in using an effective design process in which their natural behaviors were supported with explicit features. Most importantly, they were able to use the grid to partition their design effort, scraps to create diagrams with easily manipulable elements, and, occasionally, the palette to reuse impromptu design languages. Most used the grid only for working with content across multiple spaces, but some benefited from the palette as well, such as when they reused scraps on multiple canvases. Calico’s strengths also include supporting the individuals in focusing on each other during the design session, as shown by my conversation analysis.

The tradeoff to Calico’s support for informal design is that the resulting designs remain
sketches, and cannot be analyzed or used in ways that more formal design tools support. This kind of tradeoff is a recognized problem with informal tools, and overcoming it is becoming an active area of research [61]. Another observed weakness is that Calico’s interface is so different, that it takes users significant time to become familiar. Many of its benefits require the user to think ahead, otherwise they may not be using Calico optimally. Scraps in particular have a delayed pay-off, which led some pairs to not use them since they were not needed in the moment. Finally, Calico’s single user input limits how a group of people may use the tool. While it improves focus, it can effectively become a bottleneck to expressing ideas if the participants want to temporarily work in parallel, a behavior which I saw put to both good and bad use in the whiteboard pairs.

4.5 Threats to validity

Several threats to validity exist that I must keep in mind when considering my results. First, with respect to construct validity, the participants were limited to a single two-hour time span and had no access to others. Real design typically happen over longer periods of time and involve many different stakeholders. Despite this difference, during these longer periods of time, design at the whiteboard certainly happens [15]. While my work will not support all of a design, it usefully support that slice of design that happens then and there.

Second, I must consider threats to internal validity, i.e., the factors that affect my ability to claim cause-effect relationships from the results. The single user input limitation and therefore the prevention of parallel work, is the primary concern. Exit interviews confirm this as participants in the Calico pairs felt handicapped by this limitation, and given the opportunity they would have preferred parallel work. On the other hand, results from my analysis also show that Calico pairs spent a greater amount of uninterrupted time explaining alternatives and criteria to each other, building greater cohesion. Moreover, Calico sessions
still exhibits significant similarities with the non-Calico sessions in the ways the designers worked.

Third, the conditions of the experiment may pose a risk to its external validity, i.e., its ability to generalize to what people do. In order to use Calico to its maximum potential, it would require a longer period of time than what was available in the experiment in order for people to become familiar with its features and power. Another factor was that the participants in the study were all graduate students, and so there is a chance that they may not represent the behaviors of experienced professional software designers. Novice designers have a greater tendency to become fixated on a design decision prematurely [7], and novices tend to mismanage their time by focusing on a single issue for an extended period whereas professional designers know when to move on [3]. Given that the comparative nature of my experiment focused on Calico versus non-Calico design, and not on expertise, I did not further examine this factor.

Despite the fact that the experiment has its differences with real-world design, it is interesting to note that there is a relationship with how real designers work. The study conducted by Olson et al. [58] demonstrated that a design session conducted in a controlled laboratory experiment is representative of a design session as it happens in the field with professional designers. Since my study uses the same measurement as those from [58] and also has a high correlation with their results ($r = .85$), by transitive logic, I can say that the study performed here is representative of a design session as it occurs in the field.

4.6 Summary

In this chapter, I have presented Calico Version One, a sketch-based software design tool that provides support for the targeted subset of four design behaviors of software designers
from Chapter 2 when they work at the whiteboard. Using Calico, software designers were able to fluidly create, manipulate, and explore a design problem and its possible solutions. They did so with four features, including scraps, a grid, a palette, and gesture-based input to enhance the experience of whiteboard software design. The usefulness of these features was demonstrated in a multi-pronged, detailed evaluation, with participants in the experiment using Calico’s advanced features and exhibiting the kinds of design behaviors toward which Calico was designed. Lastly, a rigorous analysis of the design conversations demonstrated that Calico not only does not interfere with the design activity, but also may lead to more shared focus between participants and longer discussions of various aspects of the designs.
Chapter 5

Calico Version Two

Equipped with the lessons from Chapters 4, I have gone through an iterative process of designing, implementing, and evaluating various possible incarnations of Calico. Through experiments and trials with the technology over several years, including using the tool in my own meetings, deploying it in the classroom [47], and formal experimentation [48], I have slowly, but surely, developed an understanding of how to mesh more advanced functionality with the fluidity that the whiteboard experience demands. I have specifically refined what first was a broad range of exploratory features into a small, cohesive set of just four features that form the conceptual core of Calico.

The laboratory evaluations in Chapter 4 have given insight into the usefulness and shortcomings of the features of Calico Version One, such as scraps, the palette, and the grid. Participants found scraps useful to manipulate content, but found them difficult to use as representations for class diagrams because scraps could not be resized or rotated. Similarly, participants found the palette useful as a global clipboard to transfer content across canvases, but the inflexible nature of scraps made reusing them from the palette difficult. The grid itself proved to be a highly used feature in navigating and organizing sketches, but
participants reported wanting more organizational power such as grouping sets of canvases into topics.

From the insights found in Chapter 4, I have arrived at the final four features in this chapter, which I believe to be effective in supporting the fourteen design behaviors identified in Chapter 2. For reference, Table 5.1 summarizes the design behaviors here in this chapter, for reference. The features in this chapter represent the final iteration that will be evaluated in Chapter 6.

In this chapter, I first describe the motivation that lead to the changes I made to each of the Calico Version One features and to the introduction of new ones. The motivation is explained in terms of the design behaviors and the final feature set is mapped to their motivating design behaviors. Next, I describe the features in detail, eliciting the exact functionality for each feature. Lastly, I describe the implementation details of the system.

5.1 Updated features

In this section, I review the changes between Calico Version One and Calico Version Two, as well as the motivations that lead to these changes. I do not go into great detail for the features here, but only refer to the features at a high level. Section 5.2 describes the features in detail.

5.1.1 Changes to basic sketching

With respect to basic sketching, Calico Version Two includes added functionality that enriches the basic sketching experience of Calico Version One. The first fundamental change to Calico is its transition from a single client application to a distributed, synchronous sketch-
ing application. One of the greatest challenges encountered by participants in Chapter 4 was the inability to perform asynchronous work. Given the constraints of the technology involved with the Hitachi Starboard FX-DUO, we chose to reimplement Calico as a distributed application so that users on separate machines could sketch at the same time. The inclusion of distributed, synchronous sketching better supports design behavior 12, in which designers switch between synchronous and asynchronous work. Several other small features were included as well, such as the ability to import images and email canvases, which help to bring outside work into Calico.

5.1.2 Changes to scraps

With respect to scraps, much of the interaction changed between Calico Version One and Calico Version Two based on our feedback in Chapter 4. First is the introduction of selection scraps (center of Figure 5.1). In Chapter 4, participants in the user evaluations often used scraps for single actions, such as moving, and then would “drop” the scrap after moving its content. Based on this usage, the behavior of scraps was changed such that newly created scraps instead function as a lasso, and will disappear unless they are “pinned” by the user, as in Figure 5.1. This behavior, termed selection scrap, increases the fluidity of using scraps by reducing the amount of actions needed to perform common actions such as moving sketches.

Second is the functionality to transform existing sketches into scraps. Participants in the user trials of Chapter 4 reported that they sometimes did not use scraps because it required them to plan ahead. That is, they did not know early on some areas would need to be an object later. Retroactively “promoting” such an area to a scrap lead to objects with double boundaries. To solve this, this second behavior addresses this issue and further supports refinement of sketches (design behavior 3) by allowing the user to create scraps when they need them without as much need to plan ahead.
Third, the menu for scraps was revised to appear as a “bubble menu” (Figure 5.1), which presents the buttons on the periphery of the scrap. Users of Calico Version One had difficulty “hitting” the menu button, which was small, and also stated that the button and menu contents occluded the visibility of the scraps themselves. Items in the bubble menu are much easier to tap on and do not occlude associated scrap contents. Furthermore, it was much easier to add new previously requested features such as resize and rotate, since the bubble menu has room to accommodate additional items. The bubble menu further replaces much of the functionality enabled by the gesture system in Calico Version One. Finally, in exit interviews for the laboratory evaluation of Chapter 4, participants reported that they did not use a number of features, because they were not discoverable through the set of gestures. The bubble menu instead uses graphical icons to offer this functionality. Gestures have largely been removed from Calico Version Two, with the exception of the press-and-hold gesture.

Fourth, connectors were revised such that they could also be created from existing sketches, matching the revised behavior of scraps, and also now retain their shape (as shown in Figure 5.1). Preserving their shape provides the user with greater control in how they express their arrow, improving their ability to draw different types of connectors (design behavior 1).

![Figure 5.1: Regular sketches can be transformed into selection scraps, which can further be made permanent by “pinning” the scrap.](image)

Additionally, specialized scraps, such as text-scrap and list-scrap, were introduced. Chap-
ter 4 demonstrated the importance of creating and managing lists of text. Participants in the laboratory evaluation from Chapter 4 devoted entire canvases to writing out requirements as plain text. Text-scraps and list-scraps target these actions specifically by automatically organizing scraps for the user.

Despite these major changes to the interactions of scraps, much of their fundamental nature remains the same in order to continue supporting the same design behaviors. Scraps still retain the shape of the pen stroke used to create them. Parenting is enabled by stacking scraps on top of one another to implicitly group them. Both of these functionalities contribute towards supporting the first and second design behaviors.

### 5.1.3 Changes to palette

With respect to the palette, its basic operation has changed little. The palette continues to serve its purpose as a global source to save scraps, which can immediately be reused anywhere by dragging it back to the canvas from the palette. However, the palette does include updates that allow the user to create multiple palettes of scraps that they can move between or delete.

The palette has, however, been greatly enriched by the other new features. The revised distributed nature of Calico allows the items in the palette to be automatically shared by everyone who is connected to the same server. The ability to resize and rotate scraps, as well as the functionality to save scraps with imported images into the palette, increase the ability to create and reuse scraps pertaining to a project. They, in particular, increase the palette’s ability to support the creation of impromptu notations (design behavior 4).
5.1.4 Removal of grid and introduction of intentional interfaces

With respect to the grid, the feature has been replaced by intentional interfaces. While the grid has proven useful in past evaluations, there remains the potential to capture the relationship between the canvases within the grid. In our experience with the grid, I saw that users typically dedicated entire rows or columns to a single topic. That behavior inspired intentional interfaces, in which users may declare their intent when creating a new canvas. Using intentional interfaces, users may declare their intent, such as “exploring a new alternative”. Intentional interfaces not only enable the user to organize their canvases like the grid, but also provide the user with greater ability than with the grid. Here I review intentional interfaces, as well as the shortcomings of the grid that intentional interfaces address.

Intentional interfaces is composed of three pieces. First, rather than a grid layout, intentional interfaces organizes canvases into a radial layout (Figure 5.2), which can further be organized by branching. This functionality was created to help provide a greater grouping mechanism for canvases. While the grid did serve as an effective and straightforward mechanism to organize canvases within a design session, in practice, the grid has several shortcomings that merited a more complex feature such as intentional interfaces. For example, the participants in the laboratory evaluation in Chapter 4 grouped canvases of similar topics, such as requirements of interface mockups, in close proximity to one another. In exit interviews, participants expressed that they desired to go one step further with explicit grouping.

In the second piece of intentional interfaces, each canvas has a tag that explicitly associates it with the previous canvas it branches out from, i.e., it is an “alternative”, “level of abstraction”, and so on, of the previous canvas. Figure 5.3 depicts two canvases associated in this manner, in which one canvas is marked as an “alternative” of another. In Chapter 4, the participants were observed to shift between perspectives of the same entity. Tags en-
able the user to declare their intention when creating a canvas, thus providing a lightweight mechanism for supporting many of the navigation design behaviors 5, 6, and 7 in Section 2.3.

The third piece is the breadcrumb bar (top-left of Figure 5.2), which allows designers to navigate through the canvases. This third feature enriches the radial layout with an additional method to navigate between the canvases, and displays the associated tags as well.
5.1.5 Introduction of fading highlighter

In addition to the previous features, a new feature, the fading highlighter, is introduced to support the explanation of sketches (design behavior 13). The highlighter separates the act of highlighting from drawing. This functionality allows the designer to point, circle, and draw sequences over diagrams, but the strokes made by the designer fade a few seconds after they are made. While a small piece of functionality, the highlighter serves an important role. It allows a group of designers, and even the individual designer, to perform mental simulations (design behavior 8) and reviews of progress (design behavior 9), both while at the same whiteboard and over the network in distributed cases. The highlighter enables users to accomplish the same purpose of gesturing over diagrams without modifying their sketches, yet be much more explicit in expressing their intent.
5.1.6 Features and design behaviors they support

With the changes and additional features introduced in this section, Calico now supports all fourteen design behaviors. Table 5.1 provides a mapping between the design behaviors and the features that support them, as well as the reason why I believe the features support the respective design behavior.

<table>
<thead>
<tr>
<th>Design Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinds of sketches software designers produce</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Continued on next page*
Table 5.1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. They draw different kinds of diagrams</td>
<td>Scraps &amp; connectors</td>
<td>User-drawn shapes are preserved; strokes drawn inside scraps are grouped implicitly; scraps &amp; connectors can be hierarchically composed</td>
<td>Scraps &amp; connectors provide a unified abstraction for informally modeling a range of notations</td>
</tr>
<tr>
<td>2. They produce sketches that draw what they need, and no more</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a. They only draw what they need w.r.t. the design at hand</td>
<td>Scraps &amp; connectors</td>
<td>Calico, at its core, acts just like a whiteboard, not dictating any content</td>
<td>Can draw just what they want and nothing more</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>2b. They use only those notational conventions that suit drawing what they need</td>
<td>Scraps &amp; connectors</td>
<td>Scraps &amp; connectors do not impose any notational conventions or uses</td>
<td>Can draw just what they want and nothing more</td>
</tr>
<tr>
<td>3. They refine and evolve their sketches over time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a. They detail their sketches with increasing notational convention</td>
<td>Scraps &amp; connectors</td>
<td>Provide additional visual structure and behaviors to scraps and connectors</td>
<td>General scraps &amp; connectors can incrementally be refined to look like and behave as different notations</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>3b. They appropriate a sketch in one notational convention into another notational convention</td>
<td>Scraps &amp; connectors</td>
<td>Visual structure &amp; behaviors of scraps and connectors can be changed</td>
<td>Scraps and connectors looking like and adhering to one notation can be changed into the look and behavior of another</td>
</tr>
<tr>
<td>4. They use impromptu notations</td>
<td>Scraps &amp; connectors</td>
<td>Any visual convention can be adopted simply by drawing similarly shaped scraps</td>
<td>New notational conventions can be introduced, used as any other, and reused</td>
</tr>
<tr>
<td></td>
<td>Palette</td>
<td>Any scrap, connector, or set of them can be stored in a palette for later reuse</td>
<td></td>
</tr>
</tbody>
</table>

How they use the sketches to navigate through a design problem

Continued on next page
Table 5.1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. They move from one perspective to another</td>
<td>Scraps &amp; connectors</td>
<td>They can mix and match different notational conventions on a single canvas</td>
<td>Different perspectives can be developed within a canvas</td>
</tr>
<tr>
<td></td>
<td>Intentional Interfaces</td>
<td>Users can explicitly request a new canvas to work on a perspective</td>
<td>Canvases with different perspectives are explicitly related</td>
</tr>
<tr>
<td>6. They move from one alternative to another</td>
<td>Scraps &amp; connectors</td>
<td>Different alternatives can be quickly constructed by copying and moving and otherwise manipulating scraps and connectors</td>
<td>Different alternatives can be developed within a canvas</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palette</td>
<td>Different alternatives can be quickly constructed by reusing elements from the palette and composing them differently</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intentional Interfaces</td>
<td>Users can explicitly request a new canvas to work on a different alternative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scraps &amp; connectors</td>
<td>Different abstractions can be quickly constructed by copying and moving and otherwise manipulating scraps and connectors</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. They move from one level of abstraction to another

Continued on next page
Table 5.1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intentional Interfaces</td>
<td></td>
<td>Users can explicitly request a new canvas to work on a deeper level of abstraction</td>
<td>Canvases with different abstractions are explicitly related</td>
</tr>
<tr>
<td>8. They perform mental simulations</td>
<td>Highlighter</td>
<td>Users can use the highlighter to mark up their diagrams without editing them</td>
<td>The design can be gestured at without modifying it</td>
</tr>
<tr>
<td>9. They juxtapose sketches</td>
<td>Scraps</td>
<td>Uses can move perspectives, alternatives, and abstractions next to one another by moving scraps</td>
<td>The pieces of the design can better be compared without looking across the canvas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Users can compare canvases by zooming in on a portion of the intentional view</td>
<td>Content across canvases can be more easily juxtaposed without manipulating content within canvases</td>
</tr>
</tbody>
</table>

*Continued on next page*
Table 5.1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. They review their progress</td>
<td>Intentional Interfaces</td>
<td>Users can step back and examine their progress and process, overall and in parts, in the intention view</td>
<td>The tags and visual structure provide some insight into the rationale of canvases</td>
</tr>
<tr>
<td>11. They retreat to previous ideas</td>
<td>Intentional Interfaces</td>
<td>Users can choose to enter one canvas in the intentional view or make a new canvas at any time</td>
<td>Users can return to older versions they kept</td>
</tr>
<tr>
<td>12. They switch between synchronous and asynchronous work</td>
<td>Intentional Interfaces</td>
<td>Users can choose to enter one canvas in the intention view, or make a new canvas and work separately</td>
<td>Work is synchronous while in one canvas, asynchronous while in different canvases</td>
</tr>
<tr>
<td>13. They explain their sketches to each other</td>
<td>Highlighter</td>
<td>Users can use the highlighter to draw attention to certain parts of a canvas</td>
<td>Draw attention of designers to highlighted area</td>
</tr>
</tbody>
</table>

*Continued on next page*
Table 5.1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Supporting Feature</th>
<th>Design Principles</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>14. They bring their work together</td>
<td>Palette</td>
<td>Designers can place sketches from different canvases into palette and later merge them into a single canvas</td>
<td>Users can return to older versions they kept</td>
</tr>
<tr>
<td>Intentional interfaces</td>
<td></td>
<td>Users can navigate back and forth between canvases, and generate new ones</td>
<td></td>
</tr>
</tbody>
</table>

5.2 Implementation

In the previous section, I motivated the changes to existing features and the introduction of new features for Calico Version Two. In this section, I explain the *how* for each feature, explaining how each feature works in detail.
5.2.1 Implementation of basic sketching

Figure 5.5 presents Calico as it first appears when a designer enters a canvas. Users can immediately draw or write by dragging their pen across the whiteboard. Just as on a standard whiteboard, they can make any marks they wish, anywhere, in any shape. The drawing canvas is largely clear of any obstructions, with features available on the periphery to maintain the appearance of a standard whiteboard. The side panels are mirrored on both the left and right sides, and a panel at the bottom displays status messages and has a minimal set of auxiliary buttons.

Figure 5.5: The Calico canvas interface, including the side panels and the drawing space.

The side panels have an assortment of features to help the user while sketching. The green
5.2.2 Implementation of scraps

As mentioned in Section 5.1, scraps enable users to both manipulate content and create informal diagrams as in Figure 5.6(e). In order to support users in these activities, scraps have developed a unique set of functionality. Scraps can be created while sketching using the scrapping gestures in Figure 5.6(a) and 5.6(b), which transform drawn content into the selection scraps in Figure 5.6(c) for quick manipulation of sketches, or into the regular scraps in Figure 5.6(d) for representing diagrams. Once made into a regular scrap, it becomes an implicit group that is universally manipulatable, stackable, and relatable, as in Figure 5.6(e). We review these functionalities one by one.

Scraping gestures

Calico has two basic gestures that it uses for creating scraps directly while drawing. The first gesture creates a scrap by circumscribing an area and releasing the pen in a landing zone, as seen in Figure 5.6(a). In order to not interfere with the sketching activity itself, the landing zone will only appear if the stroke is sufficiently long enough. The second gesture for creating a scrap is done by pressing-and-holding the pen inside a stroke that is already circumscribing an area, which animates a dotted red circle and creates a scrap from the
enclosing stroke, as in Figure 5.6(b). The second gesture allows the user to create a scrap after the stroke has already been created, either as a recovery mechanism if the user missed the landing zone of the first strategy, or if they choose to convert an already existing sketch into a scrap. The ability to create scraps from plain sketches helps support the refinement of sketches (design behavior 3).

Selection and regular scraps

Scraps created from the pen gestures are first made into selection scraps (Figure 5.6(c)), which disappear after the scrap loses focus. Upon creation, the selection scrap highlights any content inside of it, and allows the user to manipulate the selected contents using the bubble menu surrounding the scrap, as seen in Figure 5.6(c). When the selection scrap loses focus, it immediately disappears and returns its contents to the canvas. If the user wishes to permanently retain the scrap, they can tap any of the two scrap icons in the upper left of the bubble menu, which transforms it into a regular scrap, as in Figure 5.6(d). Selection scraps allow users to benefit from the manipulation capabilities of scraps without forcing the contents of their sketch to be scraps.

Implicit grouping

Scraps build upon the approach taken in Translucent Patches [43], which allows users to explicitly declare an area as a group. Anything that is either entirely circumscribed in the first place or otherwise written or drawn in this area afterwards is automatically part of the group. Consider the sketch of ATM in Figure 5.6(e). It was first drawn on the canvas, then circumscribed by the stylus to become a scrap. The scrap is now a persistent object with a blue background. Any further additions to the ATM scrap, or any other scrap in Figure 5.6(e), automatically become part of that scrap.
Figure 5.6: Scrap functionality.
Manipulation

Scraps are movable, copy-able, rotatable, and resizeable. Tapping or writing on a scrap immediately highlights it and presents the bubble menu on its periphery, as in Figure 5.6(c) and 5.6(d). These seemingly innocuous actions in reality represent a significant improvement over the standard whiteboard: content drawn can be rapidly reorganized. It particularly is important that such reorganization takes place in the language of the user: elements that they have deemed of sufficient importance to promote to being a scrap are the elements that are manipulated.

Stacking

Moving a scrap to a position where it is entirely overlapped by another scrap attaches it to the scrap behind it, allowing users to quickly create a stack of scraps (thereby creating hierarchically composed groups), as one would a pile of papers. For instance, the scraps labeled “Deposit”, “Withdrawal”, and “CheckBalance” in Figure 5.6(e) are part of the scrap labeled “Transactions”. If “Transactions” is moved, “Deposit”, “Withdrawal”, and “CheckBalance” are moved as well. Dragging a scrap off of another scrap un-groups it. Moving the scrap labeled “Deposit” from its current location will ungroup it from “Transactions” and, in one fluid motion, group it with “User Interface” when the user drops it there. Note that dragging a scrap implicitly moves it to the top of the order of scraps; scraps do not slide under other scraps.

Connectors

By dragging the digital stylus from one scrap to another, the pen stroke becomes highlighted and presents the user with an icon to transform that stroke into a connector. Alternative,
as with scraps, connectors can be created retroactively by press-and-holding a stroke that begins and ends in a scrap, and then tapping the “create connector” button. The connector preserves the shape of the stroke, but is decorated with an arrowhead. The connector is persistent and anchored to the places where it originated and ended. When scraps are moved, the connectors move accordingly and keep the two scraps related. In Figure 5.6(e), the scrap “ATM” relates to the “Transactions” and “User Interface” scraps, and the scrap “Transaction” relates to “BankDB”.

**List-scrap and text-scrap**

List-scrap, depicted in Figure 5.7, are a specialized feature that organize scraps into a linear vertical list, and changes its boundaries to fully contain the children within it. If a regular scrap has children, the user may promote the regular scrap to a list-scrap. Promoting a scrap to a list-scrap reorganizes the items contained in it to be a list. Just as with implicit grouping of regular scraps, the user can drag scraps on top of the list to automatically group them. The list reorganizes items after inserting the recently added scrap into the vertical list. Each item in a list has an associated box, which can be checked and unchecked. Further, as with regular scraps, lists can be nested within other lists to create multi-level structures.

Text-scrap were designed to work in conjunction with list-scrap to help the user create content more quickly from the keyboard. Using either the text-scrap button on the menubar (Figure 5.5), or by pressing the enter key while in the canvas, the user can create a new scrap using their keyboard. If a list-scrap is already selected, the new text-scrap will be automatically appended to the end of the list.
5.2.3 Implementation of palette

The palette saves a template of a scrap that can be reused (See Figure 5.8(a)). Each scrap contains a palette button on the top right of its bubble menu (Figure 5.8(b)), which adds that scrap to the palette bar. Once added, the user can reuse the scrap by pressing the pen on the image of the icon that they wish to use, and dragging it back onto the canvas. The items on the palette bar are globally available both across all canvases in the grid, and all users connected to the same server. Further, the palette has multiple sets that the user can switch between using the up and down arrows on the palette bar in Figure 5.8(a), or can also create a new set by pressing the plus icon. This enables users to manage multiple related sets of symbols. The palette bar contains several other features for managing the palette, such as creating a new empty palette, deleting a palette, and importing images into a palette. When the user is finished using the palette, they can toggle it to be invisible from the bottom menu bar, as seen in the bottom right of Figure 5.5.
5.2.4 Implementation of intentional interfaces

The intentional interfaces feature presents a novel method to organizing sketches across canvases. I break the intentional interfaces feature down into the following components: the wall that organizes topics, clusters that organize canvases into a radial pattern, tags that related canvases and organize them into branches, the canvas interaction with tags, and the breadcrumb bar that aids in the navigation between canvases.

The wall perspective

The intentional interfaces feature is organized into three levels: the wall, clusters, and canvases. The wall provides a high level mechanism for partitioning canvases. While the grid is no longer a feature within Calico, the grid can be thought of as being equivalent to clusters, and the wall as one level of organization higher. The inclusion of the wall is motivated by long-term, multi-month usage of Calico, in which multiple people were using the same Calico grid to manage their projects. For example, Figure 5.9 depicts the wall belonging to the SDCL group, which contains content that is several years old. The wall and clusters provide a convenient way to separate projects between people. In other long term usages of Calico, users that shared a Calico grid requested a separate space to work with different topics. In the wall, grouped content is segregated from other grouped content. In the grid, one person’s “space” of canvases can cross with another persons space of canvases. Intention interfaces
eliminates that problem, helping both with navigation of canvases and people working together in the same design space. Further, canvases can now be labeled, providing additional support to organizing clusters.

Figure 5.9: The Calico wall belonging to the SDCL group, which contains sketches that are several years old and are used by several group members regularly.

From the wall, users may navigate and manage their canvases. The user may enter a cluster by tapping on the white space within a cluster, or directly into a canvas by tapping on a canvas. They may also move canvases between clusters by dragging a canvas onto a different cluster. Clusters are created and deleted automatically, where a new cluster is created when there are no more empty clusters, and extra clusters are deleted such that there remains only one empty cluster at a time.

The cluster perspective

Within a cluster, canvases are organized into a radial layout. The radial layout provides a method to organize a scalable number of canvases, where the size of the cluster is increased
to accommodate more canvases. Figure 5.10 depicts the cluster perspective of “Nick M.’s stuff”, which is the first column, second row cluster in Figure 5.9. New canvases are added to a cluster by tapping the “new canvas” button in the top right of Figure 5.10. In order to help the user in mentally organizing the set of canvases in a cluster, the organization of clusters takes its inspiration from a mechanical clock. The canvases are sorted such that their order ascends in a clockwise pattern, where the first canvas is always located at the midnight position of a mechanical clock-hand, and the most recently added canvas will always appear at the end of the clock-wise radial loop, just before the midnight clock-hand position.

Within a cluster, the user may move canvases freely. In preliminary feedback of Calico Version Two, users reported that the cluster provided them a meta-design space, and organizing the positioning of the canvases was part of their design process. In order to accommodate this action, canvases that are moved from the radial layout become “pinned”, and will maintain their spatial position in the cluster regardless of other canvases being added or deleted from the radial layout. Canvases may further be “unpinned”, in which case they return to the
radial layout and their positioning becomes automatically managed again.

**Tagging**

Canvases in the radial layout may branch outwards from the center with the use of tagging, as depicted in Figure 5.11. In the laboratory evaluations of Chapter 4, the users often moved to a new canvas for a particular purpose, such as exploring a new perspective, diving into more detail, or exploring an alternative. Intentional interface tags formally capture these links between canvases. Upon visiting a new canvas, the user is asked to tag the new canvas with an intention (Figure 5.12). After choosing a tag, the new canvas is linked to the previous canvas in the radial layout. The tagged canvases form a linked list in the radial layout, thereby preserving their relationships with one another. For example, Figure 5.11 depicts a diagram that is refined over several canvases using tags. The canvas in the center of the circle, “9. MiSE - Figure 2 original”, contains the original diagram that was imported as an image. The work from the center canvas is extended into “9.1 MiSE - Figure 2 revised”, which is tagged as a “Continuation” of the previous canvas. The diagram is refined and revised until the final diagram was reach in canvas “9.4 MiSE Figure 2c”.

Tagging is introduced in order to address the “neighbor knowledge awareness problem” [22], which was observed from participants usage of the grid in Chapter 4. In the neighbor knowledge problem, canvases that are initially located adjacent to one another typically have related information, but that relation can be lost if the canvases are moved away from one another. With respect to the grid, users typically cluster related content together, but, due to fixed spatial constraints, the grid does not scale and frequently must be manually reorganized. The radial layout in the cluster perspective, in contrast, does scale, and preserves the relationships between canvases.

The tag panel as presented in the canvas perspective is depicted in Figure 5.12. The tag
Figure 5.11: A diagram is refined and revised across several canvases, which are linked with “Continuation” and “Alternative” tags.

Panel is initially populated with a set of tags inspired by design behaviors 5, 6, and 7, such as “alternative”, “perspective”, and “abstraction”. The user, however, may add, edit, or delete the set of tags. Tags can be selected by tapping on the tag itself, after which the tag panel is dismissed. The user may toggle the tag panel at a later point by pressing the tag button in the status bar (bottom right of Figure 5.12).

Navigating canvases and breadcrumb bar

From the canvas perspective, the user can create, manage, and navigate their canvases within the context of intentional interfaces. In past evaluations of Calico, users often wanted a new empty canvas for a certain purpose, but did not necessarily want to go to the grid to find an empty one. Intentional interfaces reduces the cognitive burden of the previous version by removing the necessity of choosing the new canvas’s location. Instead, the user creates a new empty canvas by clicking the “new canvas” or “copy canvas” buttons (depicted in Figure...
Figure 5.12: The tag panel appears in the upper-right upon entering a newly created canvas. 5.5), which each take the user to a new canvas, and in the latter case, populate the new canvas with a copy of the content from the original canvas at which the user was working.

The user’s recent navigation history is also recorded to help them navigate canvases. The user may use the backwards and forwards buttons (depicted in Figure 5.5) to navigate among the history of visited canvases without returning to the cluster or wall perspectives. When the user returns to the cluster or canvas perspective, the most recently visited canvas is highlighted with a blue halo (as in Figure 5.11).

The breadcrumb bar at the top of the canvas, depicted in Figure 5.13, further helps the user in navigating between their canvases. Visible in both the canvas and cluster perspectives (but not the wall perspective), the user can tap on a canvas to reveal its associated canvases. From these canvases, the user can further tap again to directly navigate to any canvas, cluster, or the wall. For example, in Figure 5.13, the user has tapped “9. Mise - Figure 2 original”, which revealed a drop-down that displays canvases 9.1 through 9.4 (all of which
Figure 5.13: The breadcrumb bar in the upper-left enables users to navigate between canvases without using the wall or cluster perspectives.

are also depicted in Figure 5.11.

5.2.5 Implementation of highlighter

The fading highlighter enables the user to draw on the canvas without modifying its contents. The fading highlighter creates a dark-golden stroke that is 3px wide, and begins to disappear after 4 seconds until the stroke created with the fading highlighter is deleted. The strokes created with the fading highlighter are visible to everyone connected to the same Calico server and using the same canvas, and the stroke will also disappear and be deleted in the same intervals on remote machines. The fading highlighter supports users in performing mental simulations (design behavior 8), reviews of progress (design behavior 9), and explaining sketches to other designers (design behavior 12), particularly if the other designer is participating in the discussion over a network in a remote location.
5.3 Examples

In this section, I demonstrate how I envision the features to support the design behaviors in practice by presenting examples of their use. All examples are hypothetical. Note too, that these examples do not represent the full range of how the design behaviors may occur when used in actual work, but they provide an approximation of how the features in Table 5.1 will support their respective behaviors.

As an example of how these features support design behavior 1, examine Figure 5.15. Shown is a sketch that includes a mix of different representations, including an approximation of a UML class diagram, a user interface mockup, and a list of user stories. Scraps and connectors provide the building blocks to compose class diagrams and relate them to one another, and stacking scraps on top of one another represents containment within a class diagram. The user interface mockups are also sketched on top of scraps.

To show design behavior 2 as performed with scraps and connectors, I remodel the example from Figure 2.4 in Chapter 2 in Calico, as shown in Figure 5.16. Most scraps, such as “Intersection”, “Car”, and “Analysis” contain only minimal use of notations and low detail,
Design behavior 3 (they refine and evolve their sketches over time) is performed using gestures to refine normal sketches into scraps and connectors. The example in Figure 5.17, for instance, shows the progression of components from plain sketches into scraps and connectors. In the far left of Figure 5.17, the items begin as a brainstormed list of elements in a model. In the center of Figure 5.17, the designer temporarily uses scraps (no longer visible) to create space between the components by moving them. They thereafter add add a bounding box, just using plain sketching, to each element, which is a notational element they are borrowing from UML class diagrams. Later in their session, the designers reach a point in which they wish to move the elements around as part of their exploration, but maintain the arrows between their components, and thus convert their sketches into scraps and connectors using the press-and-hold gesture, as shown in the far right of Figure 5.17.
Design behavior 4 (they use impromptu notations) can be performed using scraps and the palette to save and reuse visual icons. Consider, for example, the intersection scrap previously shown in Figure 5.16. On the right side of the same figure, the designers shift their attention to the map, in which they design the interaction of cars between many intersections. In order to save effort in re-sketching elements, the designers shrink the intersection on the left and add it to the palette, and now use it as an intersection icon in their map. The designers do the same with other elements, such as cars and roads, generating an impromptu set of elements for their map.

For design behavior 5, moving between perspectives, designers use a combination of scraps and intentional interfaces in order to move between canvases focusing on different content. As a canvas becomes full with sketches, the designer creates new canvases. For example, the designer in Figure 5.18 transitions from focusing on the UML class diagrams in the center of Figure 5.18, to then focusing on generating lists of requirements (shown on the far right). They then use the intentional interfaces view, depicted on the far left, to zoom in and view different perspectives next to one another. As the designer generates many different
Designers may use scraps and connectors to refine their designs from plain sketches into representations that more closely approximate UML class diagrams.

Perspectives, they use the intentional interfaces view to manage and navigate between them, particularly tagging the relationships among the canvases.

To navigate between alternatives, design behavior 6, designers primarily use intentional interfaces in order to move between canvases, but they may also use scraps and the palette as well. Figure 5.19 shows two alternative sets of activity diagrams. In the center canvas, the designer copied and pasted an activity diagram created in a formal modeling tool into Calico. Using this as a reference, the designer copied the canvas, and explored different configurations to improve its performance. In order to do so, the designer drew three different activity diagrams using scraps to model nodes, and the palette to generate multiple nodes.
quickly. Upon finishing each activity diagram, the designer added each to the palette, and used the saved activity diagrams as a template to explore further alternatives. The final set of activity diagrams is shown in the right canvas.

Figure 5.19: The intentional interface view (shown far left) provides a birds eye view of the two alternative canvases to the right, one which is formal (center), and the other which is informal (right).

In order to illustrate how designers move from one level of abstraction to another, i.e., design behavior 7, Figure 5.20 depicts a traffic simulator map drawn across two different levels of abstraction. Shown on the left, the intentional interfaces view shows both the center and right canvas, tagging them both as abstractions of one another. As in the example for design behavior four, the intersections in the right canvas were generated using scraps and the palette to copy sketches from the center canvas into the right canvas, in which the traffic simulator is designed from the perspective of a map, as opposed to individual intersections. Both the center and right canvases originated from the same set of sketches, but were resized and repositioned to emphasize the intersection (center) or the entire map (right).

Figure 5.20: Intentional interfaces helps designers in moving between perspectives using tagging and organizing spaces.
For design behavior 8, designers use the fading highlighter in order to mentally simulate over a sketch. Depicted in Figure 5.21 are the aggregated markings of the fading highlighter over the UML class diagram from Figure 5.16. The annotations show the passage of data between the class diagrams, and the logic used by the components.

Figure 5.21: The fading highlighter helps designers in performing mental simulations by allowing them to sketch over their diagrams without marking them.

In many of the preceding examples, scraps and intentional interfaces inherently helped the designers juxtapose their sketches, which is design behavior 9. By zooming and panning in intentional interfaces, the designers juxtaposed related canvases in Figures 5.18, 5.19, and 5.20. While the layout of the canvases in each example was done automatically by Calico, designers may juxtapose any two canvases in the intentional interface view by manually dragging them next to one another. Within canvases, scraps may also drag sketches next to one another to compare them.

For design behavior 10, consider a situation in which the same designer who created each of the designs in the preceding examples wishes to review them. One method is to tap the breadcrumb bar in their current canvas in order to move between canvases. Another method is to navigate using the cluster view, zooming and panning between the different canvases in order to mentally summarize the work they have done. The links and tags between canvases, as in Figure 5.18, further help guide the attention of the designer while reviewing.
When retreating to previous ideas, design behavior 11, designers use intentional interfaces view to navigate to and recover old sketches. Intentional interfaces supports this design behavior by allowing the designer to create as many canvases as they need, and further relieves them of needing to erase past work. Intentional interfaces provides multiple ways of grouping canvases, such as grouping canvases into clusters, and by linking canvases using tags. These methods of organization help the designer in navigating back to old content.

For design behavior 12, consider Figure 5.22, in which the top two screens are a pair of designers working together and the bottom screen is the second designer at a later point in time. Initially, the two designers work together synchronously in designing a database. While the left designer designs the system architecture connecting a client to their database, the right designer simultaneously creates a list of elements in the system architecture. After creating the list, the second designer believes that another approach may be better, and breaks away to the canvas at the bottom of Figure 5.22 to work asynchronously and sketches a new design alternative.

![Figure 5.22: Designers may work together synchronously by occupying the same canvas, or asynchronously by working in different canvases.](image)

In design behavior 13, designers use the fading highlighter to explain their sketches to one
another. For example, a designer (visible in the left of Figure 5.23) is explaining the plugin architecture of a software system to a colleague, whose view of the sketch is shown in the right of Figure 5.23. While explaining the passage of information from “Server” to “CalicoEventHandler”, the designer double underlines “Server”, and draws an arrow in the direction that information is passed. The yellow strokes are immediately visible on the colleague’s laptop on the right of Figure 5.23, but disappear after a few moments. As the designer continues the explanation, they circle, underline, and draw arrows between components to supplement their verbal explanation.

Figure 5.23: The fading highlighter helps designers in explaining their sketches to others.

When bringing together their work, i.e., design behavior 14, designers use intentional interfaces and the palette to merge their content. With intentional interfaces, they copy the contents of their canvas, and refer to the contents of their previous canvas in order to merge the content. With the palette, they merge their content by copying scraps from several canvases onto a single canvas, and generate a new design on the final canvas.

5.4 Implementation notes

Calico Version Two’s implementation consists of approximately 100,000 lines of code written in Java 7.0. Calico was developed using the Eclipse development environment, and was tested on Hitachi FX-Duo Starboard interactive whiteboards. It is portable across several
operating systems (i.e., Windows XP/Vista, Linux, and Mac OSX), though all of my labora-
tory evaluations described below were performed using a Starboard connected to a Windows
machine that ran Calico. ASUS EEE121 tablets were also used.

5.4.1 Architecture

The architecture of Calico Version Two is based on a client-server architecture. A single
server instance manages the content of all canvases, all clients connect to the server transmit
inputs from their respective user, and the server broadcasts all updates from clients to all
other connected clients. The server is strictly a headless application that listens for clients on
a pre-determined port, and establishes a synchronous TCP connection all connected clients.
In order to reduce the amount of network bandwidth used during usage, Calico Version Two
mostly (but not strictly) uses smart clients, in which much of the computational work in
Calico, such as geometry for resizing and rotating scraps, is done on client machines and the
results are sent to the server. In our own usage, a single Calico server instance is capable of
handling up to 20 users sketching simultaneously before performance is severely impacted.
The aforementioned scenario involved usage by students in a classroom scenario.

In order to manipulate the state of the canvas, Calico Version Two uses a command pattern.
An individual command consists of a byte array, which is formally called a CalicoPacket
within the architecture. The CalicoPacket is Calico’s own binary format for transmitting
information, and is more compact than other more verbose messaging protocols such as
XML and JSON. Both the client and the server contain a matching master list of all possible
commands called NetworkCommand class. In order to perform a command, such as a user
drawing a pen stroke, a CalicoPacket is created that contains the pen stroke command, the
command is interpreted by a singleton command interpretation class, which converts the
CalicoPacket from raw byte array data to usable values, and performs the command. The
CalicoPacket is used to send and receive information from the server.

The client uses an architecture that is inspired by the MVC design pattern, with some slight variations. The client interface is built on top of the Piccolo2D architecture. In this framework, we build on the construct **PCanvas** to represent the container of all elements, and **PNode** to represent all objects on a PCanvas. In our architecture, both the PCanvas and PNode classes serve the purpose of both the model and the view. There is a further set of controllers that manage manipulating the model, and painting the representation of the model to the view. Each model element may always be represented as a CalicoPacket.

The core set of classes that form the content in Calico are **CCanvas**, **CStroke**, **CScrap**, and **CConnector**. The container for all sketches is represented by CCanvas, which itself inherits from PCanvas. A CCanvas may contain any sketch element, which includes the set of elements that extend PNodes. These elements that extend PNode are CStroke, CScrap, and CConnector. A Scrap may contain both strokes and other scraps. A connector has a one-to-two relation with scraps, in which the head of a connector is associated to a scrap and the tail another.

In order to manage input from the pen, a visitor design pattern is used to accommodate a flexible amount of modes (drawing, erasing, and highlighting), with a flexible number of unique objects. The interface for all objects that are traversed in the visitor pattern include: **actionPressed**, **actionDragged**, and **actionReleased**. When receiving input from the pen, an input controller first processes the input to determine the appropriate handler that implements the traverser pattern, then forwards the event to that handler. Both the canvas and the scrap implement the visitor design pattern to uniquely handle input. When the visitor design pattern visits the canvas handler, a stroke is created on the canvas. When it visits a scrap, the stroke is drawn directly to the scrap. A benefit of using a visitor pattern for handling input is that future plugins may provide new input handlers to Calico. For example, the scrap-list was first added as a plugin and uses a custom handler for organizing
elements.

The architecture further contains a plugin design pattern supported by an event handler and event dispatcher. The motivation behind the plugin architecture of Calico was to partition developer contributions to Calico’s codebase. By modularizing Calico’s components, several individuals could more easily contribute code to Calico without creating conflicts each others’ contributions. Both the palette and intentional interfaces are developed as plugins to the system. In order for a plugin to extend the system, a plugin may register itself as a listener to specific commands. In turn, when a command is performed by the command interpreter, the event handler broadcasts the event to all plugins, which may choose to operate on that action.
Chapter 6

Evaluation

In the previous chapter, I presented the final version of Calico with features that address all fourteen of the design behaviors, as well as examples of how the features may support the design behaviors in practice. In this chapter, I present an evaluation of Calico Version Two’s ability to support those design behaviors through an “in the field” qualitative evaluation. In this chapter, I simply report the experiences from those who used Calico Version Two in the course of their own work and map their use onto the design behaviors. In Chapter 7, I reflect on these experiences.

Within the evaluations of Chapters 4, I presented the study of software designers in a controlled setting, designing a solution for a hypothetical problem. This setting provided the evidence to suggest that scraps, the palette, and the grid can indeed support a subset of the fourteen design behaviors. However, in moving forward to evaluate the final version of Calico, it is prudent to consider the shortcomings of an evaluation conducted within a controlled environment. First, it is possible that the short duration of the laboratory studies does not give participants sufficient time to become accustomed to Calico’s features. Longer-term use will allow groups to receive more practice, and learn how to use the features more effectively.
Second, group behaviors such as switching between synchronous and asynchronous work and explaining their sketches are more pronounced in groups of three or more. Design behaviors based on group activities do indeed happen with only two designers, but the support Calico provides may have a more noticeable impact on collaborative work in larger groups than two. Third, Calico may have other benefits for group projects that may not appear in our analysis framework. For instance, Calico may help group meetings start more quickly because the sketches persist between one meeting and another.

Therefore, in this chapter, I report on a longer-term, qualitative study of Calico in use at software companies to not only evaluate its support for the fourteen design behaviors, but also understand its role within the context of a real-world design environment. In order to conduct this evaluation, I performed a multi-week qualitative study at three sites, including a commercial open source software company, an interaction design company, and a distributed research group. In this chapter, I report on my findings of how Calico is used over the long term at these three settings, how users incorporated Calico into their own work, and how it affected the way they design.

After reporting on each site, I report on the design behaviors as they occurred collectively across all three sites. My focus is to examine if the design behaviors occurred while using Calico, and if so, what features supported those behaviors beyond how they would manifest themselves on a regular whiteboard.

The rest of this chapter is organized as follows: Section 6.1 describes the setup of the study, the participants, and the data collections used. Sections 6.2 - 6.4 presents the results of the field studies framed within the research questions proposed put forward in Section 6.1 and Chapter 3. More specifically, Section 6.2 describes the findings at the commercial open source software company. Section 6.3 describes the findings at the interaction design software company, and Section 6.4 describes the findings with the distributed research group. Section 6.5 collectively reviews the three field sites for evidence of design behaviors and how they
were supported by Calico’s features.

6.1 Method

I evaluated Calico at three field sites, including two commercial companies and a geographically-distributed research group. At each site, I verified that a Calico installation was setup correctly, which included a large electronic whiteboard, a server instance of Calico, and access to pen-based tablets that could also access Calico. All participants were trained in the use of Calico.

Figure 6.1: The physical setup at the commercial OSS company.

The first group is a commercial open-source software company, located in southern California, that develops software within the healthcare industry. In this text, I collectively refer to the members of this company as the OSS group. A software team of five people within
the company, including four software developers and one team lead, used Calico in their project. In this deployment, depicted in Figure 6.1, a Hitachi Starboard FX was installed on site, as well as three ASUS EEE121 tablets for use with the whiteboard. Within the same space, there were two couches present in front of the board, and a regular whiteboard next to the couches. All group members were given a tutorial on how to use Calico Version Two, and shown how to launch Calico and connect to the server from their own machine. For the duration of the study, the team was engaged in architecting and implementing the next version of their project, which was written using Java software. The board itself was physically adjacent to their desks. This group was evaluated over a four week period.

![Figure 6.2: A tutorial of the usage of Calico given to the interaction design group.](image)

The second group was an interaction design firm located in northern California. In this text, I collectively refer to the users from this company as the interaction design group. In this deployment, depicted in Figure 6.2, a Hitachi Starboard FXDUO88 was installed on site. Upon installation, the members of the company were invited to a tutorial and shown how
to launch Calico from their own machines and connect to the boards. While no tablets were installed, as was done with the OSS group, the company did already have pen-based tablets which could launch Calico. Within the company, two individuals used Calico to support their work. In contrast to the OSS group, the individuals at the interaction design company were not responsible for code, but instead used Calico to perform interaction design activities, such as creating user personas using notes from interviews and creating storyboards. The pair of individuals used Calico over a five day period to process their interviews.

The third group was a distributed software research group located at the University of California, Irvine on the west coast and Carnegie Mellon University on the east coast. In this text, I refer to this group as the research group. This deployment, depicted in Figure 6.3, mixed existing users of Calico at UC Irvine, and people new to Calico at CMU. Two members

Figure 6.3: The physical setup of the research group.
were located on the west coast and had access to two adjacent Hitachi Starboard FXDUO77 machines, as well as tablet based machines. One member was located on the east coast with access to an HP tablet machine with an electronic pen. This research group collaborated on a research project for a massive online development environment for developing Javascript projects. This group was tasked with designing the front end and writing the software for their project. While they used Calico for a long period of time, only the most recent seven months are evaluated. During that time, the CMU member visited UCI for a week of intense work with Calico; though it saw frequent use thereafter. While apart, skype was used to coordinate their weekly meetings, and was used alongside Calico.

6.1.1 Data collection

The observations presented in this chapter are primarily based on data collected from interviews and usage logs generated by Calico. The evaluation period for each site was at least four weeks, and the participants at each site were free to use the system as much, or as little, as desired. Given the long duration of the study, and that usage of Calico by users at each site was opportunistic rather than planned, we did not use video.

In order to review the design activity performed using Calico, I examined the usage logs produced by Calico. Each action taken by users was recorded into a history file, including all drawing activity, navigation, usage of features, and accessing content from remote machines. Each user action included a timestamp, name of the machine that was running the Calico client, action performed, and an image of the canvases at the time the action was performed.

Based on examinations of logs and the content drawn, I conducted semi-structured face-to-face interviews. I conduct two semi-structured interview with the OSS group, one with the interaction design group, and one with the research group. In each interview, I began by asking “What was your most vivid design experience with Calico?” From this question,
participants typically walked through the designs they created in Calico, physically pointing, mimicking their usage of features, and describing the contents of, as well as the activity leading to, their designs. During these walkthroughs, I asked for clarification of content within their designs as well as activity that I observed from their usage logs.

In subsequent questions, I specifically targeted the first two research questions. First, in order to evaluate whether Calico could be considered “minimally invasive”, I asked participants if they needed to make any compromises in using Calico, what obstacles or surprises they encountered in creating their designs and if they struggled with the features. I also inquired how design sessions would have gone had they not used Calico.

Second, in order to evaluate if Calico has a “coherent set of features”, I asked participants to report their experience on using the features in general, what features they found helpful, and if the features clashed with one another.

Due to the sensitive nature of some of the data collected, measures were taken to ensure that data was securely collected. Of the three sites visited, those in charge of two of the sites (the OSS group and the research group) provided consent to use the data collected for academic publication. The third site (the interaction design group), however, was under a strict NDA agreement, and all images, logs, and raw data were not permitted to be taken off-site. In this case, usage logs were processed and reviewed on-site, and only personal notes were permitted to be taken off-site. Furthermore, for the purpose of publication, relevant images containing sensitive content were redrawn such that key insights, such as usage of notations, are preserved, but the content itself is anonymized.
6.1.2 Data analysis

Based on the logs and interviews, collected data from each session was reviewed in two phases. First, I reviewed interviews for each group for responses pertaining to the overall design activity that occurred, their general feedback on their experiences using Calico, and feedback on specific feature usage. In the second phase, I reviewed both the usage logs and interviews and categorized my observations into the following three categories: (1) how they navigated between canvases, (2) the representations they used, and (3) how they conducted group work (mimicking the three categories of design behaviors).

The results from the three field sites are partitioned across Sections 6.2, 6.3, and 6.4 due to the variety of behaviors observed. The observations across all three sites are aggregated in the overarching discussion presented in Chapter 7.

6.2 Results: Deployment at a commercial open source software company

Of the entire duration that Calico was installed at the OSS group, the developers reported using Calico in three extensive design sessions. Across all three sessions, the developers were engaged in developing the next version of a healthcare “message processing” tool.

6.2.1 Overview of design sessions

In the first design session, depicted with excerpts of the canvases in Figure 6.4, the developers turned to the whiteboard to sketch out a user interface for creating custom handlers for different types of messages. In the second design session, depicted with excerpts of the
canvases in Figure 6.5, the developers turned to the whiteboard to design a set of slides that will be used to explain the software architecture of their system. In this case, all members already understood the architecture, but wanted to create a representation “that was easy to understand”. In the third design session, depicted in Figure 6.6 with excerpts, while coding, a developer turned to Calico in order to refactor source code that needed to be updated for their upcoming software release.

In the following, I group my observations of the developers’ usage of Calico based on how they navigated between canvases, the representations they created, and how they used Calico to manage group work.

6.2.2 Feature usage

General feedback

On the whole, the individuals that did use Calico reported that they did not find the features to be invasive to the basic sketching activity. At a bare minimum, all individuals said they were able to use the large electronic whiteboard as they would a regular whiteboard. Individuals reported some drawbacks with the hardware itself, in which the electronic whiteboard hardware experienced lag so they needed to write slowly to create legible representations. However, the pen-based tablets were much more precise. They also reported some surprises in the usability of the tool, for example they expected the forward and backward navigation buttons to step through the chain of linked canvases in the intentional interface, instead of the visited canvases. Also, the members of the OSS group expected greater functionality in changing colors of scraps and text-scraps, but fell back to plain sketching to use colors.

When asked if they made any compromises in order to perform their activities using Calico, users responded that they did not need to make many compromises. The only compromises
(a) Exploration of entities using box-and-arrow diagrams

(b) Hierarchical perspective of component structures from Figure 6.4(a)

(c) Refined mock-up for user interface in Figure 6.4(a)

(d) Alternative user interface mockup to Figure 6.4(c)

(e) Mock-up of use cases created after discussion of Figure 6.4(d)

Figure 6.4: Representations used by the OSS group in their first design session to design a user interface.
Figure 6.5: Representations used by the OSS group in their second design session in which they worked on representation that details the software architecture for handling different types of messages.
Figure 6.6: Representations used by the OSS group developer to refactor their code in the third reported design session.
that they reported were writing slower and larger on the electronic whiteboard. If they did not use Calico, they reported that they would have moved to a meeting room with several whiteboards rather than the space immediately adjacent to their desks. One participant reported that in one activity they would have written on the window using erasable markers, but instead chose to use Calico. They also reported that they normally would have taken pictures of their work, but instead used the email feature of Calico.

Feature specific

When asked about the features as a whole, the members of the OSS group reported that they did not feel that the features conflicted with one another. Additionally, both interviews and usage logs confirmed that all features were indeed used at least to some extent, with the exception of the palette, which was very rarely used. The following captures the general feedback from the OSS group concerning each feature.

Basic sketching and features. Aside from the issues with the responsiveness of the large electronic whiteboard mentioned above, OSS group members reported that the basic sketching features were (e.g., changing pen color, changing line width, undo, and redo) helpful. Sketching on regular tablets was responsive, and developers from the OSS group stated that synchronous sketching was helpful in meetings, allowing members to work together, or hand tablets back and forth between each other while designing with the large electronic whiteboard. They reported frequent usage of multiple colors and the undo/redo functionality. They did, however, mistake the “clear canvas” button for the “eraser mode” button several times, leading to the canvas being accidentally cleared, but this action could be undone with the “undo” functionality. Two of the developers in the OSS group reported some difficulty adapting because upon using the device they default to gestures they use on other devices such as pinch-and-zoom, which are not available in Calico. Another member of the OSS group requested additional colors be made available as well.
Scrap. The participants found the use of scraps helpful on occasion. One member from the OSS group wanted to create text-scrap with different colored text, but instead turned to writing by hand instead of typing when this was not possible. Further, when using text scraps at the large electronic whiteboard, members from the OSS group found it cumbersome to switch between the bluetooth keyboard and the large display, and stated that they would have preferred an on-screen keyboard. Writing content was sloppy, so where possible, they created text scraps to make content more legible. Otherwise, OSS group members reported that scraps were sufficiently helpful to move content and represent objects. Among the three design sessions observed, scraps were used to represent objects in two of those sessions, and in the other session, were used simply to manipulate plain sketched content. In another design session, one of the members of the OSS group encountered difficulty in using the “shrink-to-contents” functionality for scraps, which creates a rectangular scrap around sketches if present, or simply a rectangle if no content was present. The OSS group member reported wanting to create a rectangle, but the “shrink-to-contents” button instead created a smaller rectangle around the existing sketch. Lastly, a member of the OSS group requested that scraps be resized with a corner anchor, without locking their aspect ratio.

Palette. The palette received light usage by the OSS group. The palette was used in one of the three design sessions, as a few scraps were used across several canvases. When asked how often they reused scraps, the OSS group collectively replied “rarely”. They reported that they typically went to a new blank canvas when discussing their software system, after which they archived the canvases used by emailing them to the entire group. With respect to reusing scraps with the palette, they stated that it was faster to simply redraw their representations.

Intentional interfaces. OSS group members responded that they found the “chaining”, i.e., linking canvases using tags, as supported by intentional interfaces to be helpful in capturing individual design sessions. They used the “new canvas” and “copy canvas” buttons
several times with tagging to create chains, and all design activities were done in a single cluster. When drawing within a canvas, one OSS group member reported that they exclusively used the breadcrumb bar to navigate between canvases. Another developer from the OSS group preferred to use the cluster view, but reported being annoyed that the cluster view changed their zoom perspective in an unexpected ways. In their words, the view “kept jumping around”. As a result, the users reported a lot of unnecessary panning in the cluster view. Two other developers in the OSS group also reported that they found the forward and backward navigation buttons confusing. They expected these buttons to navigate to the next canvas in the chain within the intentional interface, not the most recently visited canvas. One developer in the OSS group asked for a minimap of the cluster view to be directly visible in the canvas. One member of the OSS group called the organization “wizardry”, since they could not see how it changed.

**Fading highlighter.** The OSS group reported that the fading highlighter was useful within group sessions. They used the fading highlighter in two of the three sessions, and reported that they used this feature to explain designs.

### 6.2.3 Canvas navigation

**Using intentional interface chains to capture design sessions.** The OSS group reported that they organized each design session by “chaining” canvases within the cluster view, as depicted in Figure 6.7. Each design session was linked together using tags, typically with either the “perspective” tag or the “alternative” tag. Each new design session began as a new canvas within the center of the radial circle, and extended outward using canvases linked as chains. A member of the OSS group reported that panning through their chains of canvases was helpful in reviewing the sketches that they created.

**Using canvases to explore perspectives.** The OSS group reported that they used sev-
Figure 6.7: Cluster view of canvases. The three major sessions were grouped by “chaining” canvases.
eral canvases to explore their designs across multiple perspectives. They explored different perspectives both by using different types of representations, such as user interface mockups and software architecture diagrams, and also by creating sketches that depicted different parts of the system.

In their first design session, depicted in Figure 6.4, they created sketches using different types of representations to explore the components of the user interface and its behavior at runtime. In Figure 6.4(a), they used box-and-arrow diagrams alongside low-detailed user interface sketches to brainstorm major components of the user interface and their relationships to one another. In Figure 6.4(b), they shifted to a hierarchical list to record the structure. In Figures 6.4(c) and 6.4(d) they depicted the contents of Figure 6.4(b) as it would appear to the end user. In Figure 6.4(e), they sketched out three different use cases, where “CH”, “CH2”, and “CH3” represent three different results for input entered by the user in the scrap “All Channels”.

In their second design session, depicted in Figure 6.5, they explored different parts of the same design across the different canvases in order to document the flow of information through the system. For example, Figure 6.5(a) shows “message processing” sending information to the “Event Bus”. Figure 6.5(c) shows how information is passed from the “Event Bus” scrap onto the “Event Listener” scraps. Figure 6.5(d) further shows how information is passed from the “Alert” scrap, which itself is an instantiation of an “Event Listener”, onto the “Email” and “Channel” scraps. The OSS group members sketched out each step in order to thoroughly explain the rules and logic used for passing information between components.

**Stepping through a design using levels of abstraction.** The OSS group used multiple levels of abstraction to aid in explaining their architecture. They used it both to progressively step into the detail of their design, and also to back away to once again see the bigger picture.

When stepping progressively into their design, they made use of the “copy canvas” feature to
repeat a canvas, but replace parts of their sketches with more detailed versions to step into lower levels of abstraction. The second design session in Figure 6.5 shows two such cases. First, Figure 6.5(b) shows a lower level of abstraction of Figure 6.5(a). Between Figures 6.5(a) and 6.5(b), “message processing” is replaced by scraps labeled as “Channel”, which represent the components that process the messages. In Figure 6.5(b), each colored connector represents messages of a specific type that are passed from a “Channel” to the “Event Bus”. Second, Figures 6.5(c), 6.5(d), 6.5(e), and 6.5(f) show a sequential step-through in levels of abstraction. The “Event Listener” scrap in Figure 6.5(c) is represented as the “Alert” scrap in Figure 6.5(d), where it is shown in much greater detail. The “AG1” scrap in Figure 6.5(d) is represented as the “Action Group” scrap in Figure 6.5(e), where it is elaborated in much more detail. The “Action Group” scrap in Figure 6.5(e) is again represented in “Action Group” in Figure 6.5(f), showing further sub components for “Action Group”. While creating these scraps, the OSS group used the palette to copy elements between canvases, such as “Event Bus” and “Action Group”.

After their stepwise exploration leading to Figure 6.5(f), the OSS group members sketched a high level picture of messages passing through the system in Figure 6.5(h). Not shown in the figure, the OSS group members created several copies of the contents of Figure 6.5(h) in order to explain how different types of messages may pass through the system. In their session, they returned to previous canvases when they had questions about particular components.

**Copying canvases to generate alternatives.** The OSS group sometimes used multiple canvases to create alternative solutions to existing designs. For example, a member of the OSS group reported that, while working with their team members on the user interface in Figure 6.4(c), they “did not quite agree with the design” and deviated from the group to create their own version. To do so, they used the “copy canvas” button, selected the “alternative” tag in the intentional interface tag panel, and created a new user interface, the result of which is shown in 6.4(d).
In a third design session, one of the OSS group developers used multiple canvases to help them in iterating on past designs. After finishing the design in Figure 6.6(a), the OSS group member actually implemented the designed changes in the software, to later return to the large electronic whiteboard to verify that their source code correctly implemented the original pseudocode in Calico. In this second iteration, they created a copy of the original canvas and pasted a screenshot of the source code that resulted from the previous design session, where they continued to refine the pseudocode.

### 6.2.4 Representations

The OSS group created several types of representations in their design sessions.

**Box-and-arrow diagrams.** Across the design sessions, they created what they termed “box-and-arrow” diagrams. They did not make any direct reference to notations such as UML class diagrams or ER notation, but rather viewed their sketches as rudimentary abstractions of their existing software system. The first was used for brainstorming, and the second for explaining an architecture.

In the first design session, depicted in 6.4(a), they created a set of box-and-arrow diagrams to brainstorm the components of a user interface. The design session began by first listing out the relevant entities in the software, which were the text scraps “Source” and “Destinations”. In this session, the OSS group experimented with different combinations of actions, which were reflected with further text scraps associated with connectors at the bottom. They used connectors to represent the data sources that the ‘Source” and “Destinations” scraps could pull from and send to.

In the second design session, the OSS group used a combination of scraps, connectors, and plain sketches to represent software components and the passing of data. When asked why
they sometimes represented components as plain sketches and other times using scraps, they reported that they used scraps to represent components that they were actively designing and using. The OSS group reported that elements in sketches, such as “In” and “Out” in Figure 6.5(a) and LLP in 6.5(h), represented outside information. Creating items such as “Event Listener” in Figure 6.5(c) allowed the components to be reused and moved. For passing data, they simply used colors to represent particular types of data that were passed. For example, between Figures 6.5(a) and 6.5(b), the OSS group used four different types of colors to show the types of data that are passed from a “Channel” to the “Event Bus”. In Figure 6.5(c), the OSS group repeated the use of colors to show how the different types of data were handled.

**User Interface Mockups.** Across the sketches in Figure 6.4, the OSS group used a mix of plain sketching with scraps to design a component and its interface across several canvases. In both Figure 6.4(a) and 6.4(b), they sketched the software entities alongside a user interface mockup. In Figure 6.4(a), the OSS group reported that they created three low-detailed user interface mockups while “throwing ideas out on the board”. They iterated over their user interface across multiple canvases, two of which are depicted in Figures 6.4(c) and 6.4(b).

**Lists.** The OSS group used lists in two instances. First they used lists as a method to hierarchical organize the user interface in Figure 6.4(b). Second, they used lists as a mechanism to describe the functionality of a software architecture in detail in Figure 6.5(g). The OSS group reported that they used lists to summarize their design sessions and to reflect on designs they recently created. However, they did not create many lists because they found that the board reduced the aesthetic quality of their writing.

**Use case sketches.** In order to capture the behavior of their designs at runtime, they created primitive use case scenarios. For example, in the final canvas of their first design session, Figure 6.4(e), the OSS group sketched the behavior of the user interface when
different channels were checked. This representation depicts the results of three different use cases for the interfaces shown in Figures 6.4(c) and 6.4(d). The left scrap in Figure 6.4(e) represents the input panel, and the three scraps on the right hand side each represent the result of a different use case, in which either “Ch 1”, “Ch 2”, or “Ch 3” is tapped on the left scrap. A member of the OSS group created these three previews by drafting an initial scrap on the right hand side, and copying it multiple times, adjusting the values for each respective “Channel” selected.

In the second design session, they created several copies of the contents in Figure 6.5(h) to explore different use case scenarios. They used different use case scenarios to walk through input messages of different types and frequency, as well as using different event listeners to handle those messages.

**Source code.** In the third design session, one of the members of the OSS group used the large electronic whiteboard to help him in refactoring his code. Depicted in Figure 6.6, the OSS group member needed to refactor his code in order to process an XML file that contains new fields in the next version of their software tool. In order to carry out his design session, he opened Calico on his own desktop, copy and pasted a screenshot of the XML code that he needed to process, and continued his design session at the large electronic whiteboard. In his exit interview, he needed “a space to think freely”. While sketching, he used several custom annotations and colors to depict the process flow of different software components through the XML structure. Black vertical bars were used to represent a call stack, which he drew to help him in understanding how far the callstack goes in a recursive call. The red arrows were used to represent how the call stack moves through the data.

Overall, the OSS group member reported that using different colors helped him see things “at-a-glance”. He stepped back from his pseudocode several times, experimented with walking through the code in different orders, and used a different color for each walkthrough. The final design of the first iteration is shown in Figure 6.6(a). After implementing the design in
the code base, and upon discovery that an issue was not addressed in his first iteration, the OSS group member copied and pasted the latest Java source code to once more step through their design again to address the discovered issue.

6.2.5 Collaborative work

Preparring work ahead of time. The first design session was a shared group session that spanned the greater part of a working day. In the morning, one of the OSS group members took a tablet to their desk and worked out a design, and later presented his design in a group meeting with four other developers in the OSS group, where they continued to work together on the design for several hours afterwards. A subset of the sketches produced in this design session are depicted in Figure 6.4. In this design session, they were designing how to handle data as it is passed through each “channel”.

Explaining designs to the group. In the second design session, the team members made heavy use of multiple tablets, as well as the fading highlighter. When the group meeting began in the afternoon, the OSS group made heavy use of the fading highlighter to explain the behavior of the system shown in Figure 6.5(a). Having worked out much of the design independently earlier that day, a member of the OSS group used the fading highlighter to explain the state of the design thus far.

Spontaneous asynchronous work. During the design session, some of the design members would privately move to another cell, create a design, and call other members of the group to visit their cell. For example, while the group discussed the data that was passed from “Event Bus” in Figure 6.5(d), one of the OSS group members wanted to understand how the “Alert” component worked in the greater context, and moved to the canvas depicted in Figure 6.5(h) to sketch a much more abstract representation to show how data flowed through it. After sketching it, they called the other designers over, and made frequent movements
back and forth between these canvases to compare them.

6.3 Results: Deployment at an interaction design company

Of the entire duration that Calico was in use at the interaction design group, the interaction design group reported that they used the board on occasion for notes, and used the system to conduct one major on-going design activity. Usage logs indicate that the design session took place across five days, where usage went on throughout the work day. The two members of the interaction design group conducted the design together.

In order to preserve confidentiality in this section, the topic and content of the material have been obfuscated, and the images have been reproduced. I use generic references without mention of the target domain, and have recreated the images to align with this hypothetical task, but still represent what they drew.

Also, it is important to note that in this group, during their five day design session, they used an earlier version of Calico Version Two that did not yet include the intentional interfaces feature, but instead included the grid. The interaction design group did, however, use the intentional interface feature later in a one hour design session. All experiences reported for this group should be assumed as using the grid, unless the intentional interface feature is explicitly stated.

Overview of design session

The design session that took place involved processing a set of fifty interviews that the interaction design group conducted, and building a set of user personas based on those
interviewed. The interaction design group reported that they typically would have processed the interviews by printing the faces of the fifty people interviewed onto note cards, and working with the note cards in a physical space. They reported that they could have simply written names, but found speaking to images of the people they interviewed to be more evocative and effective. In the large open space, they would organize the note cards by physically grouping them together according to emergent categories, writing notes on the back of the note cards, and taking pictures of the groupings.

With Calico, they saw an opportunity to perform the same activity, but preserve work that would have been lost every time they re-arranged the note cards. Instead, they performed this activity by importing images of the fifty individuals that they interviewed into the palette, and dropped these onto the canvas as image scraps. In Calico, they viewed the work they did as non-destructive and preservative, where a particular arrangement of the image scraps could be copied and reorganized an indefinite amount of times. If inspiration struck them regarding an earlier arrangement, they were always able to return to a previous arrangement of the image scraps and continue with the most recent work later.

### 6.3.1 Feature usage

**General feedback**

Overall, the participants reported that the work they performed in the system was useful, but ultimately found it difficult to adapt their work habits to the structure of the system. Representing images as scraps provided advantages such as reducing the overhead in creating physical printouts for each person they interviewed, reusing them across several canvases, and allowing them to return to past explorations. The interaction design group, however, reported that they had a culture of using OneNote, a paginated sketch-based application, on pen-based windows machines, which led to a clash between the expected functionality
of Calico and Calico’s actual functionality. For example, they expected functionality such as OneNote’s lasso mode for selecting and moving content, and subsequently found Calico’s scrapping functionality confusing and difficult to adopt. Also, they found Calico’s Grid and Intentional Interface system of managing canvases not as fluid to use as OneNote’s system of vertically scrolling through paginated sketches. Further, they reported that their ability to write legibly was limited by the large electronic Starboard hardware, which resulted in larger content that was sketched more slowly than expected. They would have preferred the ability to rest their wrist on the board to write comfortably. Ultimately, however, they worked around this limitation to perform their design work.

The interaction design group reported that using Calico did indeed cause some changes in how they designed. First, they reported that, while different from OneNote’s paginated system, they felt the sensation of a greater amount of free space to sketch, and as a result, created more sketches and content than they would have otherwise. Within each canvas, however, they reported desiring more free space, either through zooming and panning of content, or paginated spaces such as within OneNote.

Additionally, the interaction design group had several feature requests. They found the use of modal dialogs, such as email confirmation dialogs and text entry, to be disruptive during their design sessions. They asked for highlighting functionality that is more analogous to its real-life counter part, i.e., persistent and shading the area behind a text with color, which they reported as using often for tagging items in their sketches. They further reported the sudden switch between sketching an object, and creating scraps to be disorienting, and requested separate modes for drawing, creating, selecting, and moving scraps. They further found the set of graphical icons for buttons and menus to not be straightforward, and requested additional descriptions.

Despite these issues, they produced twenty one canvases of content. Sixteen of those canvases made significant use of scraps, six of which used made use of multiple list scraps. One canvas
even used connectors between scraps.

**Feature specific**

**Basic sketching and features.** Participants reported some difficulty in accurately sketching with the system. They reported that it was difficult to produce legible handwriting. For example, when attempting to use print, Calico would produce cursive text. Other times, Calico would lag and sometimes lose details of their writing.

**Scraps.** The participants reported that the functionality that scraps enabled, such as moving sketched contents, was helpful, but reported some difficulty in using scraps. The participants reported that the interaction for moving scraps felt slow, particularly when moving scraps to categorize them in tables or quadrants. Further, the system began to slow down when using a large number of image scraps, thirty or more, within the same canvas. This is partly why they requested separate modes.

It is interesting to note, though, that early versions of Calico Version Two had such separate modes, which was received with very mixed reactions when deployed to students in a software design class. Students frequently had the “mode switch problem”: they did not know which mode they were in and made mistakes as a result. This was one of the reasons we reduced the number of modes dramatically.

**Palette.** The participants found the palette helpful for loading pre-existing graphics. The participants imported all fifty images of the people they interviewed into the palette, and used the images from the palette across several canvases. However, they found it difficult to find items in the palette when they loaded all fifty images into it, as the palette displays saved scraps as small snapshots, making them difficult to discern.

**Grid / Intentional interfaces.** During the majority of their usage, the interaction design
group did not have intentional interfaces, but instead had the grid. They reported that they often moved to the grid perspective in order to get a high level perspective of their work. One of the designers reported that “it’s important for me to see everything at once”. The interaction design group used intentional interfaces for a brief period, but had difficulty adjusting to using it as a method of moving between canvases. They had trouble using the breadcrumb bar to navigate between canvases, and found the preview of the canvases within the cluster view too small. This, in a way, is not surprising given the brief exposure.

**Fading highlighter.** The interaction design group reported positive feedback about this feature, however, they seldom used it in practice. They reported the feature would be highly useful in working with larger groups, but all of their sessions took place with at most two people. Further, they found the name confusing as it did not function as they expected a regular highlighter would, in which a marker changes the color behind written text. Instead, they colloquially referred to it as the “John Madden mode”, in reference to the sports announcer.

### 6.3.2 Canvas navigation

**Using perspectives to build personas.** The interaction design group used multiple canvases in order to organize their interviews by different perspectives. The people that they interviewed did not necessarily interact with the target software system the same way, but rather had different roles in interacting with the system, such as the customer or vendor, where each experienced different parts of the system. The interaction design group used various canvases for exploring the different groups, their dynamics with one another, and the emergent categories within each group. They performed this exploration by creating multiple copies of a template canvas that contained images of all fifty of the people they interviewed onto other canvases in order to explore different categorizations. For example,
they organized the people they interviewed by location, finances, knowledge of technology, and similar stories.

**Backing out to review work.** The interaction design group underwent cycles of intensive activity within a particular canvas, and bursts of consecutive movements between several canvases. In their initial exploration, they organized their image scraps along various one- or two-dimensional plots. After working in these canvases, the usage logs showed that the interaction design group moved between a zoomed out view and the different perspectives across canvases in rapid succession, which they reported as reviewing their designs in progress.

### 6.3.3 Representations

The interaction design group created several representations in order to help them categorize their data.

**One-dimensional plot.** When the designers began their design session, they created a one-dimensional plot, as in Figure 6.8(a), to categorize the people they interviewed by a particular quality. They began writing names of individuals along the plot, grouping those individuals into lists, and marked the plot’s horizontal axis with qualities at intervals between the lists of names. Additional text was written at each interval, or at edges, describing additional information about the qualities.

**Multi-dimensional plot.** In Figures 6.8(b) and 6.8(c), the interaction design group used a two-dimensional plot to further organize their image scraps. The interaction design group first began with a one-dimensional plot that depicted a range of behaviors within a particular quality. They then placed the images and names of the individuals they interviewed along this plot. Afterward, they converted the sketch into a two-dimensional plot by drawing a line down the middle, and then further grouping the images along the new plot. Within the
Figure 6.8: Various visual representations used by the interaction design group to create personas. *Note: content in sketches does not depict actual work by the interaction design group, but instead are fictional examples of types of representations that they used.*
canvas depicted in Figure 6.8(c), they juxtaposed a three-dimensional triangle plot next to a two-dimensional plot, both of which categorized a different set of scraps.

**Organization by plots, tags, and Euler notation.** After copying scraps onto the canvas, they used several techniques to simultaneously overlay different layers of categorization. Within Figure 6.8(d), the interaction design group used three types of categorization. First, they used a table. When the designers first entered this canvas, they initially dropped image scraps of those that they interviewed onto the canvas to view, and immediately afterwards drew a one-dimensional line and partitioned it into segments, similar to the canvas in Figure 6.8(a). This time, however, they extended the one-dimensional line into a table by drawing long vertical lines and categorized the scraps within these spaces. Second, they used Euler diagram notation to depict emergent groupings by circling a set of image scraps and writing the name of the category next to the grouping. For example, the two image scraps in the top left have a yellow circle around them with the text “NO CONDITION”. Third, they used color tags to denote another level of grouping. On the far left side, they created a legend with mappings between color and a category name. They then tagged the image scraps with color. The result was a set of image scraps that were organized using a table, Euler diagrams, and color tags.

**Spatially clustering objects.** In Figure 6.8(e), the designers allowed the categories to emerge from the individuals they interviewed in a bottom-up fashion. In this canvas, they first wrote the topic of the canvas, which was “Design Behaviors”, and grouped the faces of the people interviewed into categories based on their experiences, and then wrote the name of the grouping, such as “ENJOYS TO EXPLORE” or “SEEKING OTHER TOOLS”. The interaction design group first wrote the topics using the regular pen, then later converted the topics into text scraps. The interaction design group further copied and pasted their notes from that session, and annotated with a dash to indicate that it was an attribute of a cluster. The interaction design group similarly spatially clustered their image scraps in Figure 6.8(f),
but additionally used arrows, and in one case formal connectors between scraps, to associate an image scrap with more than one group.

**Story-driven flowchart.** In one particular instance, the interaction design group walked through the customer’s experience by drawing a story, as in Figure 6.8(f). From the stories they gathered in their interview, they drew the different events that occurred during an average customer’s purchasing experience, and noted possible options that may occur at each point. In order to construct the story, they reused elements they had stored in the palette, such as the image of a stick figure to indicate actors, and various icons to represent events, such as a phone to call a customer after a pending order was fulfilled. For particularly interesting events and corner cases in the story, the interaction design group placed the image of the person they interviewed to add to the story.

**Storyboards.** The interaction design group created storyboards that demonstrated user interactions in a step-wise fashion. The interaction design group drew boxes that contained a picture of each step in a chain of actions, a set of descriptions for that image, and call-outs explaining pieces of the drawn picture.

### 6.3.4 Collaborative work

**Sharing control.** The interaction design group reported that a motivator for using Calico was to share control of content on the whiteboard. They further reported that they typically fall into the roles of either producing content at the whiteboard while the other participates from their laptop and provides design critiques of the content produced. When they did use Calico, the interaction design group, for the most part, continued their traditional roles, where the non-sketching interaction designer provided critiques while developing personas. However, while developing storyboards, the interaction designer taking on the non-sketching role was pro-active in typing out notes for each storyboard frame, entering both text and
6.4 Results: Deployment at a distributed software-based research team

The research group used Calico over a period of five months in their meetings and design sessions. The high-level goal of their research was to develop an online solution for crowdsourcing the writing of software programs. Prior to using Calico, the team had already developed a software prototype, and used a whiteboard to discuss their designs, which consisted of process flows. They reported that their design had become too complex to perform solely on the whiteboard, and believed that Calico could help them evolve the design while still maintaining the informality of the whiteboard. The research group turned to Calico to “work out the details of the process flow diagram once and for all”. At the beginning of the five month period, the remote member of the team flew in locally for a five day intensive session in which they created a grand design of the entirety of the system. They continued their collaboration remotely the remaining time.

Overview of design sessions

The research group created several dozen canvases in their months of usage, using it to record meeting notes, design the underlying software architecture, and sketch out the user interface. The research group reported creating two major design diagrams, which consisted of a process-flow chart of the entire system, depicted in Figure 6.9(a), and a second flow-chart for testing user-generated code in Figure 6.9(b). Both were created during the initial five day session, and were continuously updated afterward. In creating the process flow in Figure 6.9(a), the research group first sketched out the system as it existed at the date of
the meeting. They then continued designing the parts of the system that did not exist yet. After four months of usage, they chose to perform a major refactoring of their system and the generated diagrams were no longer current. However, the research group reported that they continued to reference the old figures.

While two members of the research group were concerned with designing the system in its entirety, a third joined the research project with the intention of focusing on one specific aspect of the project, specifically how debugging is performed by the end-users of the system. Having joined the project after a prototype was already created, this additional group member needed to be onboarded [15]. Throughout this process, he worked with the other members to become acquainted with the project by using Calico to sketch components of the infrastructure relevant to his tasks, and further build his project on top of the existing pieces of the system. In his role, he created sketches pertaining to his part of the system (Figure 6.10(a)), and additional sketches to help him understand the system (Figure 6.10(b)).

6.4.1 Feature usage

General feedback

Overall, the research group found Calico to be a tool that supported them in carrying out their meetings. Prior to using Calico, the group used whiteboards to conduct their meetings, and would email pictures taken of the whiteboard among the group for those that were remotely located. They found that Calico allowed them to create more complex diagrams than they would have on the whiteboard. They further reported that for much of their design process they preferred to create the diagrams in Calico rather than a formal tool because they wished to maintain the informal feel of the whiteboard, and the ability to draw around the diagrams as needed. They did, however, recreate their major diagrams in Calico in more formal tools in order to document their designs.
(a) Process flow that described the main functionality of the software system

(b) Process flow representing flow of how user submitted programs are tested

Figure 6.9: Representations created by the research group during their one-week intense session.
Figure 6.10: Representations created by the onboarded researcher that were used to better understand the existing software system.
All members reported that they felt more comfortable sketching lots of diagrams because of virtually unlimited space. All group members valued having their old designs readily accessible, and with some framing of their context due to intentional interfaces. One group member reported that he wanted “to keep a history of what I’ve done, the branches that I’ve pruned. If you’re designing complex things with stages, you need to tell a story”.

One of the members reported that it served as a lightweight thinking tool for him. The third reported that he typically used lightweight note-taking tools such as Evernote, which helped his individual thinking process, and later shared his work with the group. He reported that Calico offered more flexibility in what he could create, and found the flexibility helpful in preparing content before sharing it in group meetings.

They reported some difficulties with Calico. Group members reported that they sometimes had difficulty launching Calico, or setting it up on a new machine was sometimes an obstacle with collaborators. Also, while sketching was adequate, the drop in writing quality was sometimes distracting. They often overcame this by using text scraps, but this was not always the case.

**Feature specific**

**Basic sketching and features.** The research group reported that the basic sketching features were adequate for their design sessions, although they reported that the electronic whiteboard made their handwriting messy. They further reported that some of the menu buttons were not clear in their functionality. They expected the “navigate backwards” and “navigate forwards” buttons to move back and forth in links within the intentional interface rather than navigating the history of visited canvases. They further thought the “clear canvas” button was the eraser mode button, causing them to accidentally clear their canvas, which they recovered by pressing “undo”.

157
Scraps. The research group made frequent use of scraps across their design sessions. They found scraps and connectors useful because both made otherwise static elements interactive, such as when working with process flows. They reported feeling less afraid to create complex diagrams because the elements could be moved. They could create space by moving scraps, and also categorizing objects, such as when placing them in tables, like in Figure 6.10(a).

They encountered some difficulties using scraps. Scraps were difficult to select when many were overlapping. Scraps could not have their text changed, which required the research group to delete scraps and their connectors when text needed to be changed. They further desired the press-and-hold selection gesture to trigger more quickly.

Palette. The research group did not use the palette in their sessions. Rather, when using repeated elements, they copied canvases and deleted non-relevant content. They also did not use repeated elements in their sketches, but rather always sketched new content, or reused old sketches.

Intentional interfaces. The research group made heavy use of the breadcrumb bar to move between canvases. They found that using the breadcrumbs was the easiest way to move between canvases. They reported that canvases were too small to visually identify in the cluster view, where canvases in a circle with thirty or more canvases became very small. Identifying clusters by name in the breadcrumb bar was easier in comparison.

Two members of the research group also did not use tagging with intentional interfaces because they found the system of tagging to be confusing. They found grouping canvases by clusters helpful because it allowed them to separate content between members. However, within clusters, they did not link canvases. The third researcher, in contrast, used tagging to link canvases into linear chains, each of which represented a topic. He reported that many canvases by themselves were difficult to return back to and read without context, but if viewed one after the other, one could more readily understand the concepts in juxtaposition.
to the previous canvas. On the other hand, he found the act of tagging canvases as “alternatives” or “abstractions” awkward because he was not sure what those meant in the context of his sketches.

**Fading highlighter.** The members made light use of the fading highlighter. They reported that, while it was useful on occasion, they preferred to talk out loud and make pointing gestures using their hands. In remote sessions, the group member from CMU relied on video communication to see these non-verbal cues. Though they found the fading highlighter helpful, they oftentimes forgot that it was available. They reported that they would have used it more if it had been easier to invoke.

### 6.4.2 Canvas navigation

During sessions involving the design of the process flow in Figure 6.9(a), the research group reported that they seldomly moved to another canvas. Rather, they used additional tools such as Google Docs to record requirements and design decisions, or simply talked out loud. All sessions using Calico also involved the use of Skype and Google Docs.

**Using perspectives to support onboarding.** The research group member who underwent onboarding used canvases of several perspectives in becoming acquainted with the software system. This member used a mix of user interface mockups, freehand sketches, tables, and other diagrams to step through the implementation of the software already written and the design of their own new features. This initial exploration involved importing an image of the interface into Calico, and stepping through the interface to inspect items pertaining to debugging user generated code. From these sketches, he switched to other canvases in which he generated the process flow in Figure 6.9(b), which was another perspective of Figure 6.9(a), but distilled to only include details relevant to his project. Later, while again exploring the process flow in Figure 6.10(a), the group member created several copies
of the canvas, preserving the process flow at the top of the canvas, and creating different tables, lists, and mock-ups below while experimenting with different algorithms.

**Exploring alternative process flows.** The same researcher that underwent onboarding drew several alternatives for his part of the system. He created several copies of the canvas in Figure 6.10(a), which consisted of a process flow diagram juxtaposed against a table. Between these alternatives, the process flow depicted at the top of the canvas did not change. Instead, the table below served as a template to experiment with alternative paths for components. The group member frequently referenced his previous canvases to compare against other alternatives.

### 6.4.3 Representations

The research group primarily used Calico to work on process flows, however they additionally used other types of representations.

**Lists.** Not depicted, lists were used on occasion to record bullet points from meetings or during brainstorming. The research group used a mix of both written lists, which were used as scratch notes, and text scraps, which were rapidly created during brainstorm sessions. On the whole, lists were not used often because the research group took notes in Google Docs, and used Calico to supplement written documents.

Additionally, in some complex diagrams, the research group reported keeping track of ideas using lists. For example, in a sketch similar to Figure 6.10(a), a group member reported writing down questions he had about states of the diagram, which he would investigate later. Similarly, the top of Figure 6.10(b) contains additional notes and questions while brainstorming a user interface.

**Timeline.** Not depicted here, timelines were used to record target dates for important
dates, such as conference deadlines.

**Process flow.** The research group used process flows extensively, as the design of how tasks flowed from one person to another are critical to their design. As such, a lot of detail was put into their process flows in order to capture the details of the system.

The research group used process flows as their representation of choice to design their system. During their intensive five day meeting, they attempted to model the entirety of their software onto one canvas, calling it a “a view onto their system”. As the design grew, they encountered issues with available space within the canvas, but did not wish to break up their design across multiple canvases. The research group reported that they “thrashed about how much detail they should put in the diagram”. As their design grew, they shifted from using plain sketches with drawn arrows, to using scraps with connectors. They found text scraps easier to read than their own handwriting and more space efficient. Also, they found the ability to have connectors remain attached to scraps as they were moved very useful, as it allowed them to make space for more states, or place similar states in close proximity to one another. After the five day session, they reported a resistance to changing the diagram because of its complexity. They “knew where everything was because it had always been there”, and when they did make additions, those parts of the diagram “grew organically” by occupying vacant white space without displacing the positions of the scraps around it.

They encoded additional information into their diagrams by using color, as depicted in Figure 6.9(a). One research group member stated that they “needed some way to encode what was going on”, where some scraps represented states, some represented tests, and some as points of branching. For example, black connectors represented a state transition with no information passed, blue represented the passage of tokens, and so on. The research group reported some difficulty in recalling the meaning of all colors for connectors. They further distinguished the scraps representing states that were not yet implemented by tagging them with color. In the upper right of Figure 6.9(a), they wrote “Red: not yet implemented” in
red, and tagged scraps with this quality by underlining them with red.

When the research group described the diagram, they saw it not as a complex network of individual states, but rather as a set of higher-level structures that represented different parts of a complex work flow. That is, they implicitly grouped parts of the diagram into larger components. Their discussions took place over different regions of the diagram, which they understood among themselves to represent components and verbally referred to them by name, but did not record within the process flow because they were not sure how to represent them. When talking about features they requested a way to declare “these are the typical paths” or “these two paths are similar”. When their discussions took place, they discussed specific workflows within the diagram, and discussed what happens in those workflows. The research group struggled with how to abstract away these workflows, and created additional diagrams, such as diagram depicted in Figure 6.9(b).

**Table.** One of the research group members used tables to help him process sketches while onboarding. He created tables such as the one depicted in Figure 6.10(a), and juxtaposed them against simplified sketches of process flows in order in support of a detailed examination. The process flow in Figure 6.10(a) contained several paths that could be taken, and the group member created tables to capture a particular workflow. He did so by listing each state in the row header, transitions along the column headers, and placing scraps in the intersecting cells. He created several such tables to capture different workflows.

One of the group members reported that they created the table in Figure 6.10(a) to examine a set of corner cases. The table served as an aid to help them think through the process flow. The group member reported that they used the table to simulate an algorithm, thinking through “how this would be executed”. They reported that scraps were helpful in being able to move things. They reported that they normally would have created this diagram on a whiteboard, but in comparison, Calico provided “lots of space”, and made their table “very clean” because they did not have to redraw things.
One of the group members, in contrast to the others, used Calico more frequently than the others to help him design while working alone. He would prepare several canvases of sketches prior to a meeting, sketches including lists, tablets, process flows, etc., to explain his designs. He expressed that he preferred to make his designs within Calico both because the other members were already using Calico, and because sharing sketches in Calico was faster because it was already a shared space.

**User interface mockups.** The research group created prototypes of the user interface late in the design cycle, after having already created a working system that they could interact with. They used mockups during discussions to suggest changes to the user interface, and when designing new web pages. During these sessions, they imported screenshots of their current interface into Calico as a visual reference, and sketched interfaces without the use of scraps.

The onboarded group member created user interface mockups within Calico in order to help him in thinking through his design. In his task, he used a combination of screenshots and sketched mock-ups to first walk through how existing end-users perform their task and submit their finished work. From these sketches, he created a process flow to capture the interaction of the user and the flow of information.

**Spatially clustered text scraps.** In two meetings, the research group generated a set of text scraps as part of a brainstorm, and later grouped them into similar topics. They did not use formal arrows, but rather used grouping symbols, such as Euler diagrams, and drew arrows between the groups to indicate different types of relationships.

**Dendrogram.** One researcher created a horizontal dendrogram to explore a set of questions generated from other sketches. The researcher began with a set of four questions, and from those questions, expanded into a tree of questions to explore a problem. They drew arrows to link items between the questions, as well as several freehand annotations.
Source code. In several of the meetings, one of the group members pasted screenshots of source code into Calico to discuss his implementation. He pasted both pseudo code and actual Javascript code. In meetings, he drew call outs from the screenshots to explain each method call, as in Figure 6.10(c).

6.4.4 Collaborative work

The research group conducted meetings using the two electronic whiteboards as well as their own laptops. During co-located meetings, they sometimes loaded both boards with Calico, but visually viewed different canvases on each whiteboard, or used the second board to display Google Docs. They reported that only one person usually sketched, while another took notes on in Google Docs, and any remaining participants simply talked.

Helping remote members “feeling connected”. In distributed meetings, remote members reported feeling “more connected” when using both Calico and video chat software. In these meetings, members often used another display loaded with either Skype or Google Hangout. The remote participant reported that the ideal setup included a camera that pointed at the boards with Calico, because it allowed him to observe the body language of the speakers. They reported that this setup was particularly helpful during the early phases of their collaboration because people often pointed to objects on the boards, gestured to content at the board from their seat, or gestured in free space while explaining an idea. The combination of both being able to see the body language of remote members and being able to manipulate that same content in real-time culminated into the sense of “connectedness”.

Continuity between design sessions. The research group also reported that Calico made it easier to stop and resume design sessions. One member reported that the “biggest benefit was crossing space and time”, in which Calico provided a consistent virtual space to conduct meetings prior to the five day visit, during, and long after. All canvases remain
in a consistent location that all group members can always access. They did not need to worry about cleaning up any materials in their space or sharing whiteboard space with other research groups conducting meetings in the same space.

**Providing an archival reference.** After the diagrams became out of date, the research group reported that they used Calico as an “archival reference”. The research group originally thought that they would continuously interact with their sketches, but instead used their sketches as visual references. The team eventually referenced the sketches less and less as they had internalized the contents of the sketches and could reference freely without the need to have the sketches in front of them. However, they reported that they valued having the sketches in Calico rather than static photographs of whiteboards. They reported that it gave the sketches “a sense of permanence” because they could return and make adjustments to the sketches at any time, which they did. After undergoing a major refactoring, the content of the sketches in Calico became outdated, however, they did reference them still at times while moving forward. They reported that it was “good to have a reference of what the architecture was like in the previous version”. They went back to past designs to see what they did before. When looking back at previous designs, they noted that “it would have been nice to have annotations as to why this [previous design] wasn’t going to work”. They remarked that while their sketched designs were most useful during implementation, they “would turn back to the diagram because they forgot how they implemented something while writing”.

### 6.5 Design behaviors

In this section, I turn to examining the design behaviors across all three sessions. I report on whether they occurred or not, and whether the features of Calico were used as intended. This is not an exhaustive review, i.e., I am not reporting on every instance of each design
behavior occurring. Rather, I am looking for proof of existence. This breaks down into three parts: (1) did I see proof of the designers performing the design behaviors, (2) did they use the features of Calico to perform them, and (3) if so, which features.

### 6.5.1 Design behaviors summary

Based on evidence from usage logs, as corroborated by the interviews, all design behaviors were observed as occurring at least once. More specifically, all 14 design behaviors occurred in the OSS group, 9 of the design behaviors occurred in interaction design group, and 13 of the design behaviors occurred in the research group. With respect to the usage of features, 13 of the 14 design behaviors for the OSS group were performed using Calico’s features, 5 of the 9 design behaviors for the interaction design group were performed using Calico’s features, and 12 of the 13 design behaviors were performed using Calico’s features for the research group.

Table 6.1 summarizes the features that supported each design behavior, which included scraps & connectors, palette, intentional interfaces, and fading highlighter. Referring back to the table of features in Chapter 5 (Table 5.1), each feature was designed to support a specific set of design behaviors. In Table 6.1, all non-targeted features are shaded in gray; that is to highlight when features were used in support of behaviors they were not explicitly designed for.

A summary of the design behaviors as they occurred is presented as follows:

1. **They drew different kinds of diagrams.** All groups performed this behavior using scraps, as well as with plain sketching as on regular whiteboards without using Calico features.

2. **They produce sketches that draw what they need and no more.** All groups
created scraps that were either sparse in details or notationally rich, depending on their needs. They also did so with plain sketching when not using scraps.

3. **They refine and evolve their sketches over time.** The OSS group refined sketches into box-and-arrow diagrams using scraps. The interaction design group refined one-dimensional plots into both two-dimensional plots and tables using regular sketching.

4. **They use impromptu notations.** Both the research group and the interaction design group created custom notations using scraps and connectors. The interaction design group created custom icons using the palette. The OSS group created their own visual language using plain sketching by creating connectors with custom colors, shapes, and textures.

5. **They move from one perspective to another.** All three groups used intentional interfaces to move between canvases of different perspectives. The OSS group and the research group benefited from intentional interfaces because chaining canvases provided an ordering for the canvases, which helped members of each group remember the meaning of their sketches within canvases. All groups also moved between perspectives within a single canvas.

6. **They move from one alternative to another.** The OSS group and the research group used intentional interfaces to copy canvases and label them as alternatives. All three groups discussed alternatives verbally without using Calico.

7. **They move from one level of abstraction to another.** The OSS group and the research group used intentional interfaces to step between canvases of different levels of abstraction. The OSS group used scraps to represent the software components they “stepped into” and “out of”.

8. **They perform mental simulations.** The OSS group and the research group used the fading highlighter to walk through the flow of data in their designs. The interaction
design group created elaborate sketches that walked through the story of a user using regular sketching, but did so without any of the advanced features of Calico.

9. **They juxtapose sketches.** The OSS group and the research group used scraps to position sketches next to one another for reference. The research group used the palette to copy content across canvases so that they could reference it. All groups also drew sketches of different perspectives in the same area to juxtapose them.

10. **They review their progress.** All groups reported that they used intentional interfaces to review the sketches they have created (and the grid in case of the interaction design group). The OSS group and the research group also created handwritten lists that summarized the designs that they recently created.

11. **They retreat to previous ideas.** The OSS group and research group both used intentional interfaces to return to past designs and design decisions. The research group also returned to designs that were several months old.

12. **They switch between synchronous and asynchronous work.** The OSS group and the research group had members that used intentional interfaces to deviate from the group discussion to their own canvas to reference content or create alternatives. However, though it did occur, it occurred relatively rarely in the research group.

13. **They explain sketches to each other.** The OSS group and research group used the fading highlighter to explain designs. The OSS group reported finding this very useful to discuss data that passed between software components.

14. **They bring their work together.** The OSS group created new canvases that combined work from previous canvases.
Table 6.1: The set of design behaviors and features that each group used to support them in the field evaluations.

<table>
<thead>
<tr>
<th>Design Behavior</th>
<th>Supporting Feature</th>
<th>OSS</th>
<th>IxD</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. They draw different kinds of diagrams</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2. They produce sketches that draw what they need, and no more</td>
<td>Scraps &amp; connectors</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3. They refine and evolve their sketches over time</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4. They use impromptu notations</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Palette</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5. They move from one perspective to another</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Intentional interfaces</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6. They move from one alternative to another</td>
<td>Scraps &amp; connectors</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Intentional interfaces</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>7. They move between levels of abstraction</td>
<td>Scraps &amp; connectors</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Intentional interfaces</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>8. They perform mental simulations</td>
<td>Fading highlighter</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. They juxtapose sketches</td>
<td>Scraps &amp; connectors</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Palette</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. They review their progress</td>
<td>Intentional interfaces</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>11. They retreat to previous ideas</td>
<td>Intentional interfaces</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 6.1 – Continued from previous page

<table>
<thead>
<tr>
<th>Design Behavior</th>
<th>Supporting Feature</th>
<th>OSS</th>
<th>IxD</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>12. They switch between synchronous and asynchronous work</td>
<td>Intentional interfaces</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>13. They explain their sketches to each other</td>
<td>Fading highlighter</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>14. They bring their work together</td>
<td>Scraps &amp; connectors</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Intentional interfaces</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 6.5.2 Kinds of sketches software designers produce

**Design Behavior 1: They draw different kinds of diagrams**

Across all three field deployments, the developers and designers created several representations using different types of notations, or approximations of notations. All groups used lists, the OSS group used box-and-arrow diagrams, the interaction design group made heavy use of one- and two-dimensional plots, and the research group used process flows and tables. Most of the canvases across all of the sites used only one type of notation, but a few of the canvases within each group also contained a mixed set of representations using different notations. This difference compared to the experiments in Chapter 4 came about because, unlike the controlled experiments in which the entire design session occurred at the whiteboard with no other materials, the designers in-the-field turned to Calico for specific tasks, such as when the OSS group created sketches to plan the explanation of an architecture, which may lead to canvases with more targeted content. Longer term use of Calico may lead to using it for...
more in-depth tasks, designers may supplement it with other tools such as word documents, their own computers, and so on. Designers may also need to create fewer sketches since they may have internalized many of the representations they sketched in previous meetings, which the research group reported to be the case for them.

The OSS group and research group created many different types of representations using scraps. The OSS group used scraps to create box-and-arrow diagrams and user interface mockups while brainstorming the elements and look of a GUI, as depicted in Figure 6.4(a), and mixed lists with user interface mockups in 6.4(b). In each group, creating the representations in the same space allowed them to evolve both concurrently, where the box-and-arrow and list representations allowed the developers to consider the components from a structural perspective, and the mockups allowed them to consider the design from the end-user’s perspective. They reported that, by depicting these elements as scraps, it made the elements easier to move and resize around the canvas, and therefore felt more like entities in their eyes. For the research group, one member of the research group mixed different types of representations in one of his canvases in order to help him think. In Figure 6.10(a), the researcher used a table in order help him step through a process flow, and used scraps to move items around the table.

The interaction design group, outside of image scraps and text scraps, seldom used scraps to depict major concepts in their canvases, but did create different types of representations as they would have on a regular whiteboard. They used scraps to depict very basic concepts, such as image scraps to show the faces of those they interviewed and text scraps to produce legible text when their handwriting itself was not legible. However, when creating complex representations, they simply sketched as they would have on a regular whiteboard. The interaction design group placed graphs of different types next to one another, such as the one-dimensional and two-dimensional plots in Figures 6.8(a), and the triangle graph and two-dimensional plot in 6.8(e).
Design Behavior 2: They produce sketches that draw what they need, and no more

All groups engaged in this behavior, in which they rarely actually completed their sketches in all detail, and used full notational conventions sparingly. Evidence of this came from both the amount of formal conventions used in the sketches, and the disparity between the design as drawn in Calico and the designs as explained in the interviews.

First, all groups varied the amount of notational conventions used between sketches, even when those sketches were of the same representation type. The OSS group, for example, expressed most of their software architectures using only boxes-and-arrows, and labeled the connecting arrows in very few instances. In most cases, the OSS group discussed the design verbally and only added detail to their diagrams in order to have something to point at during discussion. The interaction design group used varying amounts of notational convention, sometimes labeling the axes in their plots in great detail, as in Figures 6.8(a) and 6.8(f), other times including very little detail as in Figure 6.8(d). The research group also engaged in this design behavior when working with their process flows, in which only some states contained entry and exit conditions, and many arrows lacked labels.

Second, it became evident through the interviews that the sketches made in Calico omitted many important details of designs sessions. In interviews, members from the OSS group and the research group had difficulties identifying the meaning of some of the sketches they created, however they recalled the overall objective of the sketches, which they deemed more important than the individual details. These sketches were used to support activity while “in the moment”, where sketches were used to accomplish an immediate goal such as brainstorming, explaining, or, in the case of the interaction design group, categorizing people interviewed. For example, a single developer in the OSS group created a sketch to explore use cases of a user interface (Figure 6.4(e)), but only used the bare minimum detail to do
so. The sketch itself is a distillation of previous sketches, Figures 6.4(c) and 6.4(d), and does not use notations to suggest that it is a use case diagram. The source code in Figure 6.6 is also a sketch used in an individual session with minimal detail. The developer invested the time to write pseudo-code, but used arrows with minimal explanation of their meaning.

In the above cases, all groups used scraps to perform this design behavior, and the OSS group and interaction designer group used the palette as well. The OSS group reported that they used scraps to represent “important entities”, which served as a low-detail representation that they could point to during design discussions and move around the canvas. They reused these low-detail entities across several canvases using the palette as well. The interaction design group used image scraps of people, which they saw as the lowest level of detail in their sketches. They additionally saved low-detail icons such as a telephone and computer to the palette, and placed them in their sketches. Their sketches did not explain these icons, but their meaning was known throughout the design discussion. The research group also used scraps to represent states in process diagrams, which they manipulated quite heavily. Across all groups, content that was not the subject of manipulation often did not become scraps.

The fading highlighter also played an important role in supporting low-detail diagrams for the OSS group. This group used the fading highlighter extensively to discuss components, draw flows of data, and discuss details of software components without permanently adding more details. By using the highlighter instead of actually drawing more detail, they explicitly chose to preserve sketches at a low level of detail.

**Design Behavior 3: They refine and evolve their sketches over time**

Two of the three groups exhibited this design behavior. In the case of the OSS group, they first sketched the names of entities, and these sketched names eventually became text scraps.
with connectors. The result of these sketches is depicted in Figure 6.4(a). In this sketch, the developers began with simple sketches, and increased their notational convention and amount of detail over time. In the case of the interaction design group, they began with pictures of faces, and across several canvases, categorized these pictures using visual structures such as tables and plots. Within Figure 6.8(d) in particular, the interaction design group evolved their representation by beginning with a single dimensional line, adding categories on that line, and transforming it into a table. The interaction design group did not set out to create the table, but rather created the table after categorizing the faces. Other representations also began as a one-dimensional plot, such as Figure 6.8(b), which was later refined into a two-dimensional table.

The research group did not exhibit this behavior. This was because the research group began their major design sessions by copying existing diagrams from the whiteboard into Calico, and did not start design sessions from scratch, as was the case for the OSS group and the interaction design group in the sketches they refined.

The OSS group used the scraps feature to refine an existing sketch, but the interaction design group did not. Scraps helped the OSS group refine their sketches because scraps support the creation of box-and-arrow diagrams. The interaction design group refined their sketches into more complex plots, which scraps do not support.

**Design Behavior 4: They use impromptu notations**

All three groups performed this design behavior in which they created a visual language that was particular to their own respective designs. Each group made consistent use of their own devised notations, meaningfully using connectors, tagging, and self-drawn icons to layer meaning on structured diagrams.

The interaction design group and research group used scraps to create their own notations.
Both groups used Euler diagrams to group scraps by drawing circles around sketches (used within Figures 6.8(d) and 6.9(b), among other diagrams not shown). Each group tagged scraps using colors, where the interaction design group used color patches in Figure 6.8(d), and the research group used colored underlines in Figure 6.9(a). The research group, unlike the interaction design group, used color coded connectors with their scraps, which represented specific types of transitions between process flows. In the case of the colored connectors, the data types were informally known by the people who sketched them, but their meaning was not apparent from just the sketch alone.

The OSS group also created their own notations, but did not use scraps to do so. Like the research group, the OSS group used colored arrows to represent different data types. However, the OSS group improvised representations in ways that regular Calico connectors could not, such as by using shape and texture to give meaning to their sketches. The arrows in Figure 6.5(b) used a dashed texture to represent data, as opposed to solid arrows which were method calls between components. In Figure 6.5(c), the OSS group used colored boxes to represent events that were broadcasted to multiple components, but originated from the same source. Further, the member from the OSS group that created the psuedo code in Figure 6.6 improvised several types of arrows that served as placeholders for concepts during his walkthrough. Interestingly, the member from the OSS group could not identify the meaning of his own improvisations afterwards, but reported that they helped his thinking during design.

The interaction design group further used the palette to create their own set of scrap icons. They created symbols such as a phone, a computer, and other symbols to represent actions, which they used used in their storyboards by placing the icons next to actors who could perform those actions.
6.5.3 How they use the sketches to navigate through a design problem

Design Behavior 5: They move from one perspective to another

All groups shifted their focus among multiple canvases with different perspectives in their design sessions. Both the OSS group and the research group, in working with source code, moved between perspectives such as the user interface, software architecture, lists of requirements, source code, etc. The interaction design group, in contrast, used Calico to build personas from interviews, and moved between canvases that categorized the same data, but using different categorizations and visual structures (such as tables, one- and two-dimensional plots, etc.).

The OSS group made heavy use of intentional interfaces in moving between perspectives. The OSS group chained canvases in their sessions, which provided them the benefit of ordering canvases such that a chain of canvases could convey a story. Sometimes this order reflected the chronology of their exploration in the design space, where the earliest canvas in a session was on the inner circular ring. However, canvases were sometimes inserted within the chain when they returned back to previous sketches and deviated to a new idea. Overall, having the canvases linked into groups helped them remember “how the session played-out”. In some cases, members from the OSS group could not identify parts of their sketches, but were able to recall their meaning by examining the surrounding canvases in the chain.

The research group also used intentional interfaces, but to a lesser degree than the OSS group. In the research group’s intensive five day session, they reported that they preferred to maintain a single diagram so that all components of their system were always visible in their meetings. They instead preferred to examine other perspectives using their own computer within Google Docs. One of the members of the research group, however, did
use intentional interfaces to move between multiple perspectives in his own design sessions. This member used perspectives to help in learning the system. He moved between canvases containing large diagrams and others containing screenshots of the user interface in order to understand how the different pieces worked together.

In the case of the interaction design group, it was difficult to judge their performance with the fifth design behavior because they had little time to experiment with intentional interfaces in comparison to the other groups. During their five day research session, they used the grid to conduct their design session, in which they used the grid extensively to produce several different perspectives and they did move between those different perspectives, sometimes even rapidly switching between several canvases. During their one hour brief session in which they used the version of Calico with intentional interfaces in order to create storyboards, they limited their session to a single canvas, and ignored the intentional interfaces feature altogether. They reported that it was important to them to be able to see their sketches in their entirety the whole time.

While the three groups used intentional interfaces to different degrees for moving between canvases, all three groups found the “copy canvas” ability useful. The interaction design group in particular used a template canvas to begin explorations of new perspectives. For each canvas, they set a generic topic, such as “Design behaviors” (Figure 6.8(e)), from which sub-categories emerged from their grouping of content. Both the OSS group and the research group used copied content to examine elements juxtaposed against other representations, such as tables and mockups.

**Design Behavior 6: They move from one alternative to another**

Two of the groups did use multiple canvases to explore multiple alternatives, but the interaction design group did not. Within the OSS group, the alternatives were often generated as a
result of conflicting opinions during discussion. Multiple members had conflicting opinions, which inspired some members to copy a canvas and generate their own interpretation. In another case, a member of the research group returned to the same diagram of a process flow at a later point after having implemented it, and created several alternative copies of the canvas, in which he explored different process flows. Across both the OSS group and research group, the members used tags to label their canvases as alternatives. Also common in both groups, the members requested better tagging functionality to declare which alternative they ultimately chose, but also did not want to discard unused alternatives, because they valued them as records of past design explorations.

In the case of the interaction design group, they did not explore alternatives using multiple canvases. Unlike the other two groups, the goal of the interaction design group in their sessions was to build a better understanding the people they interviewed within a particular context. Instead of drawing multiple alternative sketches, they negotiated alternatives verbally, for example, by proposing different ways of categorizing content in a canvas, or if a person belonged to one category or another.

**Design Behavior 7: They move from one level of abstraction to another**

Similar to the role of perspectives, members from the OSS group and the research group moved between levels of abstraction to help focus their attention within the design. The OSS group moved between levels of abstraction in their sessions to step through the flow of data within their software architecture. In stepping through components, they copied canvases, expanded scraps to dive into more detail, and created new canvases with scraps that represent a high level view of their architecture. In the case of the research group, one member shifted a level of abstraction to break down diagrams and focus on relevant parts. He reported creating a copy of a process flow that depicted the entire system, erasing all components not relevant to his task, and designed a subpart of the system. In the case of
the interaction design group, they instead performed a bottom-up approach, working from
dozens of image scraps, and creating high level abstractions that described the people in the
photos. All observed groups created canvases with text, either handwritten or using text
scraps with list scraps, that summarized the contents of other canvases, which they referred
back to while designing components.

The OSS group and research group used intentional interfaces to move between levels of
abstraction, and the OSS group additionally used scraps for this purpose as well. The OSS
group used it the most fluidly, where they used intentional interfaces to continue their work
onto another canvas without interruption. When “jumping into” a component, they copied a
canvas, tagged the canvas using intentional interfaces, and reused the scraps that were copied
from the previous canvas. The OSS group used scraps to represent the components that they
“stepped into” and “stepped out of”. When “jumping out of” a component, they created
a blank canvas and sketched the component. The research group did not use intentional
interfaces as often, but reported it as useful for moving abstractions, and the interaction
design group did not use it at all for this purpose, neither in their sessions with the grid nor
their session with intentional interfaces.

**Design Behavior 8: They perform mental simulations**

All groups reported that they mentally stepped through their sketches, both verbally in
groups and on their own. Both the OSS group and the research group mentally simulated
the flow of data within their system. The OSS group displayed sketches of their architecture
on the large electronic whiteboard with Calico in meetings, and discussed the same sketch
for long periods of time, both gesturing at components with their hands and using the
fading highlighter from a tablet that was remotely connected to the same canvas. The
research group performed similar activities, but rather than drawing entities and software
components, they used process flows, and walked through the states of data as it traveled
Figure 6.11: The OSS group sometimes spent as long as 30 minutes walking through the data flow of a diagram using the fading highlighter.

through their system. The interaction design group, instead, walked through user scenarios, and sketched flow diagrams that described the story of the user. The interaction design group also mentally walked through the stories of the people they interviewed by sketching elaborate work flows, such as in Figure 6.8(c).

The OSS group made heavy use of the fading highlighter in this activity, the research group used it some. In one particular meeting, four members of the OSS group discussed a Calico sketch for 30 minutes using the fading highlighter, as depicted in Figure 6.11(b). The fading highlighter allowed members to use a tablet to mark up the sketch on the large electronic whiteboard from their seat, allowing them to circle items as in Figure 6.11(a). The research group also used the fading highlighter in their meetings, but not as often. They reported that they internalized their process flow diagrams, and did not need to perform as many detailed walkthroughs of their design. The fading highlighter was reported as being useful when a person had a tablet in their hands, and the screen was broadcast to everyone else. They also stated that they would have used the fading highlighter more in their walkthroughs had they remembered that the fading highlighter existed.
In contrast to the other groups, the interaction design group did not use the fading highlighter. There are two possible reasons for this. First, the fading highlighter was most useful when using multiple devices, but the interaction design group preferred to use only the electronic whiteboard. Second, it is possible that the interaction design group preferred to sketch the mental simulation as a workflow instead of using the fading highlighter. The members of the interaction design group were both skilled in sketching visual icons expressing ideas in walkthroughs, and could readily create these icons without interrupting their workflow. Figure 6.8(c) depicts an example sketch that includes such icons.

**Design Behavior 9: They juxtapose sketches**

While not performed often, the three groups did occasionally set their representations side-by-side within the same canvas, i.e., juxtapose them. In the majority of sketches created, the users preferred to create multiple canvases, and compared content informally by moving back and forth between canvases. However, they did juxtapose content within the same canvas for specific purposes, such as early exploration and reference.

The OSS group and the research group used scraps to help them juxtapose several different perspectives at once during very early phase exploration. In creating a new user interface, the OSS group created lists, box-and-arrow structures, and low-fidelity interface mockups within the same space, as seen in Figure 6.4(a). During this early phase brainstorming, the box-and-arrow diagrams evoked imagery of possible user interfaces in the OSS group members, which they sketched out in the periphery using scraps. The OSS group moved the content around the canvas using scraps, and also placed the objects side-by-side. The research group similarly used mockups alongside software structures, but instead began from the user interface and wrote pieces of code.

The OSS group and research group also juxtaposed sketches in order to have a reference
while creating a newer sketch. One OSS group member imported source code as an image to reference while designing psuedo code. A research group member also imported screenshots of the application’s user interface while designing his part of the system. He further copied pieces of the process flow and used an adjacent table to step through the diagram, as in Figure 6.10(a).

The interaction design group also juxtaposed their sketches, but did not use scraps to do so. They created multiple charts within the same space, finding that sometimes their data did not fall within one type of categorization, but two different types, as in Figure 6.8(c).

**Design Behavior 10: They review their progress**

Evidence that users reviewed their progress during their design sessions was more difficult to distinguish, but all groups reported performing this activity to some degree. Nearly all participants summarized or distilled the essence of an activity to a bullet point list in a canvas, which they would occasionally reference and update. The OSS group seldom marked up canvases with annotations, but instead created canvases with lists, in which they would note details about software components and entities from their past conversations. The research group would sometimes create bullet point lists to guide their meetings, but most often turned to Google Docs to review their agenda within their meetings. The interaction design group used a single canvas with several notes from their interviews, to which they referred when moving between different perspectives.

Intentional interfaces served a strong supporting role when reviewing progress in all groups. Usage logs showed that members from all groups would occasionally rapidly move back and forth between several canvases, or move to a bird’s eye view to review several canvases at once. Interviews confirmed that they were reviewing progress across canvases and moved to this view to “take it all in”, respectively. All groups reported that the cluster view was
not detailed enough to compare components, but instead it served to anchor discussions and allow those present at meetings to gesture at the canvases while they talked.

**Design Behavior 11: They retreat to previous ideas**

Retreating to previous ideas was a behavior that was only observed in multi-week, long-term design sessions. Both the OSS group and the research group continued to use Calico within the same project, and reported that they did not return to previous ideas until a later design session. Members from both the OSS group and the research groups reported that they created new alternative designs in later sessions, but returned to past sessions to refresh their memory on the past approaches that they took. In the case of the research group, they performed a significant restructuring of their software architecture, but one of the members reported returning to out-of-date sketches because it reminded him of the rationale for certain design decisions.

Intentional interfaces supported the OSS group and research group in retreating to previous ideas. The OSS group very rarely returned to past sketches, but they reported that although infrequent, they found it useful on occasion. The research group reported that intentional interfaces was useful to them in returning to designs that were several months old. The research group stated that they could identify their old design sessions from the cluster view based on the visual structure of canvases. The chaining of their canvases in the cluster view provided contextual cues, such as relationship of canvases and order of canvases in chains, that helped the research group members remember the meaning of their sketches.
6.5.4 How they collaborate on them

Design Behavior 12: They switch between synchronous and asynchronous work

The OSS group and the research group switched between synchronous and asynchronous work in their meetings, while the interaction design group did not. During their respective meetings, members from the OSS group and the research group deviated from the ongoing discussion to their own canvas to explore an opportunistic thought, and later share their insight with the group. The interaction design group, in contrast, have an established work culture of working in pairs to brainstorm and reflect on each others’ work. This led to the interaction design group not exhibiting this behavior.

Members from the OSS group reported that they deviated to asynchronous work at least once every design session, and did so using the intentional interfaces feature. In one instance, a member from the OSS group disagreed with the layout of a user interface in Figure 6.4(c), and used the “copy canvas” feature to begin the process of creating the alternative in Figure 6.4(d). After creating his alternative, the member presented his changes back to the group on the large electronic whiteboard. In another instance, another member of the OSS group wanted to unify the diagrams diffused across several canvases, and deviated from the group discussion to draw the system in a specific use case, which is depicted in Figure 6.5(h). Once the member brought this use case of the group’s attention, they created several copies of this canvas to examine use cases with different inputs. In both of these examples, intentional interfaces helped the group members navigate between these canvases by chaining them.

The research group also engaged in this design behavior using intentional interfaces, but to a lesser degree than the OSS group. In their design sessions, they navigated to other canvases in order to reference past sketches, and also to update sketches in other canvases. The members of this group sometimes requested the other group members to join them
in another canvas, but the remote member reported difficulty in navigating to that canvas using intentional interfaces. In contrast to the OSS group, the research group did not report branching off to create alternatives. This was possibly due to the setting. Members in this group did not have as much face-to-face time as members of the OSS group who shared an office, which led to members focusing more in meetings. The members of the research group met regularly one to two times a week, which they used to update each other on their progress, and discuss issues they encountered.

The interaction design group did not engage in this design behavior. During their sessions, one member of the group controlled the board while they worked together to address the design problem. One member of the group took the lead in creating sketches, while the other offered his feedback verbally.

**Design Behavior 13: They explain their sketches to each other**

All groups encountered situations in which they needed to explain their sketches to one another. The members from the interaction design group worked very tightly together, and most explanations came from one designer challenging the design decisions of the other designer. Members of the OSS group worked more independently, where some used Calico on their own, and later presented their designs to receive feedback from other developers in the group. The members of the research group operated much more independently, in which several days would elapse before they coordinated their efforts and would present summaries of their latest work to one another. In nearly all cases, explanations were primarily carried out by pointing, gesturing in the air, and freely speaking to one another.

The highlighter was the most helpful feature to support this design behavior, however, most teams reported often forgetting to use it “in the heat of the moment”. The interaction design group did not use the fading highlighter, preferring to simply speak out loud. The
OSS group used the fading highlighter for extended periods of time when stepping through an explanation of a design that spanned several canvases. They reported sometimes being slowed down during explanations when moving between canvases, requiring that they announce what canvas they move to so that everyone else can move to that canvas as well. The research group also found the highlighter helpful, but reported often forgetting that the feature existed.

**Design Behavior 14: They bring their work together**

Groups very rarely brought their work together, or merged their work from separate canvases, after performing asynchronous work. The interaction design group did not perform asynchronous work and did not have an opportunity to exhibit this behavior. The research group used Calico to present their work in weekly meetings, but they did not combine their work. The OSS group, however, did bring together separate ideas, but they did so by creating new canvases.

The OSS group created new canvases to combine work using intentional interfaces. In two separate occasions in which members from the OSS group moved between synchronous and asynchronous work in the same meeting, they generated copies of the same canvas that were variations of one another, but they did not merge these canvases. Instead, they created a new canvas, linking it to the previous canvases using tagging, and sketched a use case scenario that summarized the content from previous canvases.

**6.6 Summary**

In this chapter, I presented my observations of Calico Version Two in use at three locations. The commercial open source software company provided insight in how a software team may use Calico to support ongoing with work among developers while coding. The interaction
design group provided insight into how interaction designers may use Calico to help them create personas and storyboards. Lastly, the research group provided insight into how a set of distributed researchers may use Calico in support of long-term collaboration in the development of a software system for a period of several months. In the next section, I bring these separate uses together by discussing their collective lessons and insights in the context of Calico’s support for design behaviors.
Chapter 7

Discussion

In the previous chapter, I reported on the experiences of three different groups using Calico. In this chapter, I bring these observations together to answer the research questions posed in Chapter 3.

The rest of the chapter is organized as follows. In Section 7.1, I review the strengths and weaknesses of each feature, drawing upon how they were used across all three field studies. Section 7.2 takes a step back from the observations and examines the theoretical potential of Calico’s features by performing a cognitive dimensions analysis, revealing insight into why certain features did and sometimes did not provide the support that was intended. Section 7.3 then examines the overall strengths and weaknesses of Calico, reviewing the benefits and drawbacks of using Calico in a work environment. Section 7.4 discusses the feedback from the users, their personal experiences, and what they saw as important in their experiences. Section 7.5 then summarizes the contributions of this chapter.
7.1 Strengths and weaknesses of each feature

In this section, I examine each feature on its own merits to discuss their strengths and weaknesses. Table 7.1 provides an initial summary of this section, and each feature is examined in detail in the subsequent subsections.

Table 7.1: The strengths and weaknesses of each advanced feature.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scraps &amp;</td>
<td>1. Flexible nature is useful in creating many representations relevant to software</td>
<td>1. Requires training to use properly</td>
</tr>
<tr>
<td>connectors</td>
<td>2. Gestures enable quick manipulation of hand drawn sketches and scraps without mode switching</td>
<td>2. The press-and-hold gesture for selecting and moving scraps is slow for continuous arranging a large number of scraps</td>
</tr>
<tr>
<td></td>
<td>3. Existing hand-drawn boxes can be refined into scraps</td>
<td>3. Scrap color or border cannot be changed, text scraps are “visually heavy”</td>
</tr>
<tr>
<td></td>
<td>4. Bubble menu is self discoverable</td>
<td>4. No support for cardinality or annotations on connectors</td>
</tr>
<tr>
<td></td>
<td>5. Connectors retain shape of stroke, allowing diagrams to grow more organically</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 7.1 – *Continued from previous page*

<table>
<thead>
<tr>
<th>Feature</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
</table>
| Palette           | 1. Bootstraps design sessions by importing existing artifacts into multiple canvases  
2. Enables building a set of reusable icons for use in storyboards  
3. Functions as a global clipboard for copying content across canvases in order to juxtapose them | 1. Hard to find scraps in palette when it has many items (more than twenty)  
2. Plain sketches cannot be added to the palette, they must be turned into scraps first |
| Intentional       | 1. Clusters provide a simple metaphor to separate content of different projects  
2. Linking canvases allows a narrative to be constructed from a set of canvases  
3. Makes free space for sketching immediately available by allowing users to jump to a new canvas or copy a previous one | 1. Moving between the canvas view and the cluster view is not a smooth action, canvases are too small to visually distinguish from cluster view  
2. Difficult for new users to understand where a new canvas will appear in the cluster view  
3. Difficult to juxtapose content across canvases |

*Continued on next page*
Table 7.1 – Continued from previous page

<table>
<thead>
<tr>
<th>Feature</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fading highlighter</td>
<td>1. Supports explaining of sketches and mental simulations</td>
<td>1. Strokes made with highlighter are anonymous</td>
</tr>
<tr>
<td></td>
<td>2. Unlike traditional pointers, fading stroke allows for transient annotations such as arrows, underling, circling, etc.</td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.1 Scraps

Scraps, on the whole, are a relatively successful feature in Calico, and a significant improvement over their Calico Version One counterpart. Scraps replace the original lasso functionality of Calico Version One with the ability to manipulate content without changing modes. Where in the original version, one had to first create a scrap to move content and then delete the scrap, in Calico Version Two this can be done in one step with the selection scrap, which disappears after moving a scrap.

A strong quality of scraps is their flexibility in representing diagrams. While scraps are not always used in favor of regular whiteboard sketching (that is, lots of content is not “scrapified”, but stays as hand-drawn strokes), long-term users use them often and they yield benefits such as helping diagrams evolve more gracefully and arranging content smoothly. Actions that are natural when using scraps, such as annotating it with a color patch or underline, are not straightforward in other mediums. The same actions can be done on a whiteboard, but re-arranging sketches becomes more complex and requires much more
redrawing. Alternatively, one can design using a more formal software design tool, such as ArgoUML [70] or Rationale Rose [69], which allow for easy arranging of sketches, but these tools do not allow users to freely add annotations or symbols.

The flexibility that scraps afford further leads to unexpected behaviors in the experiment, which was evidenced by the interaction design group when they tagged scraps using color and the research group color coded their scraps as well.

In comparison to Calico Version One, which saw a relatively narrow set of representations created with scraps in experiments, Calico Version Two leads to design sessions with a much greater variety of representations. It is possible that a longer term evaluation provides more time to become accustomed to using scraps; however, it is more likely that the revised functionality of scraps is responsible for this improvement, simply because scraps are much easier to create and use than Calico Version One.

What scraps lack, however, is more expressive power in their visual features. Users cannot change the blue color of scraps and their outline. In the studies, users commented that they wished to create text scraps without the blue border, as they felt that it made the sketch “visually busy”. In another session, a user wanted to create text scraps with text of different colors in order to encode meaning, but instead used hand-written text in order to use colors. While these and other features may be useful, their absence does not obstruct the sessions as users find ways around them, and users remark that these sessions are provisional in that their sketches will not be published or shared with others beyond the meeting.

Scrap gestures, such as creating, selecting, and moving, are a strong and a weak point, depending on the situation. Particularly with long-term users, scrap gestures become a powerful tool to quickly move and copy content. When working with box-and-arrow diagrams, for instance, they allow users to rearrange content quickly. When working with lists, they allow individuals to adjust spacing in handwritten text or make more space. In user
interfaces, they enable rearranging of elements to simulate using the sketched interfaces. When individuals use them to categorize a large number of scraps, however, as the interaction designers did, the press-and-hold gesture to select and move scraps is perceived as too slow. The interaction designers, in this case, requested a separate mode specifically targeted at moving scraps.

A weakness of scrap gestures is also that they require some practice to become fluid. Gestures such as the landing zone to create scraps enable quick manipulation, but new users require some training before they can trigger the gesture reliably.

Scrap gestures in Calico Version Two, however, are a significant improvement over the scrap gestures in Calico Version One, which users in the original evaluations report as being unpredictable and not discoverable. In contrast, the bubble menu in Calico Version Two is straightforward and discoverable. Further, the landing zone used to create scraps is self-discoverable by users, while the equivalent action in Calico Version One is much more obtuse.

Lastly, the ability to convert regular hand-drawn sketches into scraps, i.e., refine them, further benefits designers. This addresses a significant issue found in evaluations of Calico Version One, which was that individuals often did not use scraps, because they would first hand draw a box, and later draw a scrap around the box, resulting with a hand-drawn box inside a scrap. While not used often, users at field sites did use the functionality to promote strokes to scraps and connectors for both refining existing sketches and as a recovery mechanism when they failed to trigger the “landing zone” gesture correctly.

### 7.1.2 Palette

In the evaluations, the palette performed strongly in supporting users in three scenarios. First, the palette served to bootstrap a design session for interaction designers by importing
several dozen images from outside of Calico. The imported images drove the design session which spanned several days. Second, the palette served to store a set of graphical icons that were improvised by the interaction design group in their design sessions. Members from the interaction design group found these icons useful in creating storyboards. Third, all groups used the palette to juxtapose new sketches against previously created ones. They did so by using the palette as a global clipboard to copy sketches to the canvas, so they could subsequently copy it out of the palette next to another sketch.

Some aspects of the palette can use further improvement. Users in the evaluation had a difficult time locating items in the palette when it contained in excess of twenty scraps. The interaction designers imported dozens of images of faces, which were difficult to distinguish because of their small size in the palette. Also, users wished to add sketches to the palette without first making the sketch into a scrap. Users reported that making the sketch into a scrap made it too heavy, e.g., they simply wanted the sketch without a scrap’s blue background and border.

One thing we did not see was impromptu notations specific to software code or architectures. This is not necessarily a weakness of the palette. Instead, it was caused by two factors: (1) users found it faster to redraw certain sketches, and (2) other Calico features were redundant with the palette. With respect to the first point, users often only needed the name of a component or provides a simple box with a few embellishments. In such cases, redrawing is faster than first making a scrap, then adding it to the palette, and then dragging it off. With respect to the second point, the availability of the copy functionality in numerous forms lessened the need for the palette. For example, when users sketching a software system wanted to create an alternative or move perspectives, they often used the “copy canvas” button in intentional interfaces, or used the “copy scrap” button in a scrap’s bubble menu.
7.1.3 Intentional interfaces

Intentional interfaces was a relatively successful feature in Calico. It advanced on the concept of the grid from Calico Version One, and satisfied numerous design behaviors in doing so. It provided qualities that were not available in the grid interface, such as names for canvases, relationships, and order of work performed. Intentional interfaces, however, had some weak points as well. Navigation was sometimes clumsy in the cluster view, juxtaposing sketches across canvases was difficult, and understanding the relationships between canvases was sometimes difficult for new users.

Many of the grouping mechanisms in intentional interfaces improved upon the grid interface of Calico Version One. While the grid was extremely well received by users of Calico Version One, users increasingly desired ways to separate and categorize their content. Intentional interfaces addresses this problem in several steps. First, clusters provide a generic method to divide canvases into topics, which helps partition content between different projects in Calico. Second, linking canvases provides another level of organization of canvases, while at the same time offering an ordering to the canvases. Providing an ordering was shown to be beneficial in practice as it allowed a set of canvases to compose a narrative. A sequence of canvases could show a walkthrough of a software architecture as well as the exploration of the design space across multiple perspectives, abstractions, and alternatives. Further, grouping canvases by linking them helps users better recall design sessions and understand the content of their sketches several months afterwards.

A third benefit of intentional interfaces is the immediateness of free space. In comparison to the regular whiteboard, users feel more at ease in creating more sketches because they can always create more space and return to previous sketches. The presence of both the “new canvas” and “copy canvas” buttons make moving to a new sketch a ready-at-hand action, which they use to explore different perspectives and new alternatives.
Users encountered a few issues using intentional interfaces as well. First, the movement between the canvas and the cluster view was not a smooth transition for the user. When loading the cluster view, the perspective always zoomed out, which made the individual canvases too small to distinguish. Second, users had a difficult time understanding the shape of the cluster when creating new canvases from within a canvas itself. One member of the OSS group described the shape of the cluster as “wizardry”, in which it was a mystery to him as to where new canvases would appear. Third, users could not easily compare contents between canvases.

7.1.4 Fading highlighter

The highlighter feature worked well for a particular set of scenarios. The fading highlighter stands on its own as users did not combine its use with other features, but it was useful in scenarios that the other features did not support. Particularly, it supported users in mentally simulating their work, and also in explaining concepts to others.

A strength of the fading highlighter is that it enables transient annotations in verbal walk-throughs and explanations. Traditionally, verbal explanations of sketches include either a simple pointer, like a baton, laser pointer, or finger, or the speaker must permanently mark up a sketch. The fading highlighter provides a compromise of the two approaches, allowing the speaker to use symbols that convey more meaning, such as drawing arrows between sketches, underlining them, and circling key points for emphasis.

While the fading highlighter addresses these situations well, it can still be improved to provide better support the two design behaviors that it targets. First, it could provide better support for explaining sketches by providing better awareness features. For example, the strokes should show who is making them. Second, mental simulations could be better supported if the sequence of strokes was recorded. Many of the mental simulations explain how data is
moved between components. If this explanation could be captured, and later played back, particularly with audio explanations, important design rationale could be captured.

7.1.5 Summary

Each feature brings forward a set of unique advantages not available in the other features. Scraps are strong in supporting the kinds of sketches designers draw. The palette acts as a global clipboard and makes sketches reusable across canvases. Intentional interfaces provide ways to organize, partition, and create free space for sketches. The fading highlighter supports verbal explanations of sketches. While each feature has usability issues, such as navigation in intentional interfaces or scraps requiring training to use properly, the features provide a net benefit to their users. Overall, the features showed to be understood by the users, with steady and sometimes surprising use emerging.

7.2 Cognitive dimensions analysis

Having examined the use of Calico’s features in practice, this section now examines the theoretical support of those features by performing a cognitive dimensions analysis [29, 30]. The cognitive dimensions framework exposes the affordances that a notation or medium offers. Analyzing Calico’s features through this framework provides a basis to determine how those features help or hinder the intended use of the features of Calico. In order to perform the cognitive dimensions analysis, I look at each dimension, and discuss factors that either increase, decrease, or, all things being equal, provide equivalent support for designers as compared to the whiteboard. These findings are summarized in Table 7.2.
### Table 7.2: Cognitive Dimensions Analysis of Calico

<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstraction</strong></td>
<td>Scraps help visualize abstractions and make them manipulatable</td>
<td>You can draw anything</td>
<td></td>
</tr>
<tr>
<td><strong>Closeness of mapping</strong></td>
<td>Can manipulate representations at the level of scraps and connectors</td>
<td>Can draw in any notation</td>
<td>Annotations, like cardinality, do not move along with connectors</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>Can save and refer back to previous sketches; on whiteboard they may have been erased</td>
<td>Must be socially enforced</td>
<td>Difficult to socially enforce when content is diffuse and requires hard mental operations, as opposed to whiteboard where content is all in one place</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffuseness</td>
<td>Whiteboards may run out of space and thus lose sketches that are erased; Calico never runs out of space</td>
<td>Small designs stay on one board</td>
<td>Calico breaks up the sketches across multiple canvases</td>
</tr>
<tr>
<td>Error-proneness</td>
<td>Manipulating and automated copying of existing sketches may prevent errors caused by redrawing sketches</td>
<td>No error checking</td>
<td></td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard mental operations</td>
<td>Content can be imported into Calico using image scraps; content can be positioned side-by-side using scraps; palette allows users to copy content for reference across canvases; navigation button allows user to return to previous content quickly</td>
<td></td>
<td>Design may be diffuse in Calico and details of sketches scattered across many canvases</td>
</tr>
<tr>
<td>Hidden dependencies</td>
<td>Intentional interfaces helps manage diffuseness by grouping content together to make hidden dependencies easier to find</td>
<td></td>
<td>Content may be diffuse, making hidden dependencies hard to find</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Premature commitment</td>
<td>Easier to explore alternatives by copying existing ones using scraps, palette, and intentional interfaces</td>
<td>Medium is considered provisional; rejected decisions can be preserved by crossing them out rather than erasing or deleting them</td>
<td></td>
</tr>
<tr>
<td>Progressive evaluation</td>
<td>All sketches are preserved and can be returned to; fading highlighter allows evaluating without marking the original sketch</td>
<td>Users must judge the “rightness” of a sketch by manually reviewing it</td>
<td>Diffuse designs are harder to progressively evaluate; fading highlighter may encourage less permanent marks that remind users of explanation</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisionality</td>
<td>Users can make copies and explore alternatives without losing original ideas; fading highlighter allows sketching over a design without permanently marking it; canvases can be tagged as alternative using intentional interfaces</td>
<td>Medium is considered provisional by users; anything can be changed at any time</td>
<td></td>
</tr>
<tr>
<td>Role expressiveness</td>
<td>User has unlimited space to fully express all parts of a design</td>
<td></td>
<td>Designs spread across canvases have higher diffusion, lower visibility, and may require hard mental operations to reference other canvases</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary notation</td>
<td>Secondary notations inside scraps move with scraps; custom shape of scrap can be used to capture meaning; color of connectors can be used to capture meaning</td>
<td>Users can sketch in any secondary notation</td>
<td>Annotations outside of a scrap do not move with a scrap</td>
</tr>
<tr>
<td>Viscosity</td>
<td>Scraps make sketched content not “locked in” such that can easily be moved and manipulated; gestures make manipulating scraps a fluid action</td>
<td>Users can sketch anything they want; a full whiteboard is less likely to be changed because it contains many important decisions</td>
<td></td>
</tr>
</tbody>
</table>

*Continued on next page*


Table 7.2 – Continued from previous page

<table>
<thead>
<tr>
<th>Cognitive Dimension</th>
<th>Factors that cause an increase in facility over the whiteboard</th>
<th>Factors that are equal in facility to the whiteboard</th>
<th>Factors that cause a decrease in facility to the whiteboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility</td>
<td>Sufficiently small designs are entirely visible on the whiteboard</td>
<td>Designs diffused across multiple canvases have very low visibility, leading to hard mental operations</td>
<td></td>
</tr>
</tbody>
</table>

7.2.1 Abstraction

Abstraction refers to Calico’s ability to support representations at different levels of abstraction. Since Calico at its most basic supports plain sketching, it inherently allows maximum flexibility in what can be represented, much the same as the traditional whiteboard.

Calico, however, improves on the regular whiteboard, because users can manipulate those abstractions at the level of scraps or, when using intentional interfaces, entire canvases. On the whiteboard, the designer can only draw and erase strokes. In Calico, if scraps are used to depict abstractions, it makes available the ability to quickly move the drawn abstractions, implicitly group elements, and utilize arrows that remain attached to relate the scraps. Intentional interfaces enables the user to explore different levels of abstraction across multiple canvases, and supports the user in navigating between these levels of abstraction by linking
7.2.2 Closeness of mapping

Closeness of mapping refers to how closely representations map to the referred concepts. Calico is similar to the whiteboard in this regard in that designers are at liberty to sketch any representation. If the designers decide that there is a more suitable representation, they are able to create a new sketch that depicts that representation simply by drawing its corresponding shapes. These shapes, then, can be turned into scraps when so desired.

When using scraps, Calico has the positive consequence that representations can be manipulated as objects, which may more closely map to the mental model of the designer than plain strokes would. Further, whereas individuals on the whiteboard representations may need to erase and redraw sketches in order to move them, in Calico scraps can be moved and copied without loss of accuracy.

A negative consequence of using scraps is that certain visual annotations, do not move along with scraps and connectors. This may cause sketches to become less closely mapped to their concept as cardinality annotations may become disassociated from their target sketch. This may cause additional work as users will need to manually move annotations with scraps, or simply leave the annotation behind after the scrap is moved.

7.2.3 Consistency

Consistency refers to the degree to which features of structures and syntax are used the same way throughout. Calico does not provide an automated method to enforce consistency. As with sketching, maintaining consistency between elements and sketches is up to the users to “socially” enforce [64]. If all content can be condensed to a single whiteboard, the designer
can refer back to previous sketches to check for consistency. Calico, since the designer is encouraged to use multiple canvases, it may be more difficult to manually verify consistency if sketches are spread across canvases.

However, if the design is too large for a single whiteboard, Calico would provide some help over the whiteboard. Where on the whiteboard sketches would need to be erased, the designer may save them in Calico, and refer back to them to verify consistency between old sketches and new.

7.2.4 Diffuseness

Diffuseness refers to how much the meaning of sketches is spread out across multiple sketches. Calico potentially increases diffuseness because the potential for an unlimited number of canvases may encourage designers to break up their designs across multiple canvases, something we indeed have seen with different canvases playing different roles. On the whiteboard, designers are forced to limit their design sketches to the limits of the whiteboard. If everything can be sketched on the whiteboard, then it is superior because designers can see everything at once.

However, if the design grows beyond the space of a whiteboard, Calico offers some functionality that helps in dealing with diffuseness. On the whiteboard, diffuseness becomes obliviousness, because what is not immediately visible has been erased. Calico keeps all sketches, but this raises a new issue, which is the problem of how to navigate between sketches in Calico. We designed intentional interfaces to address this issue by grouping together canvases of related content first by clusters, and second by linking canvases using tags.
### 7.2.5 Error-proneness

Error-proneness refers to the degree to which a notation induces “careless mistakes” [64]. As with plain sketching, Calico does not provide any safe guards against errors. Individuals are free to improvise and switch between notations, which may increase their tendency to insert errors since there is no clearly defined standard notation.

While Calico does not alert the user of when they have performed an error, Calico’s features do automate certain actions which may reduce the likelihood of errors. Scraps and connectors, for example, provide flexibility in creating representations by enabling the moving, resizing, and rotating existing sketches. They maintain the structure of objects as they are moved, which may help when working with box-and-arrow diagrams. In comparison to plain sketching, scraps and connectors help prevent errors by making additions to representations, like process flow diagrams, less tedious as they do not require diagrams to be redrawn when repositioning elements, as one would have to if sketching at the whiteboard. On the other hand, designers recognize that there is value in redrawing a sketch from scratch because each time a sketch is redrawn, its contents are re-evaluated [63], and copying existing sketches removes this opportunity. Errors caught during redrawing, then, may not be caught in Calico if sketches are simply copied.

### 7.2.6 Hard mental operations

Hard mental operations refers to information that needs to be referenced or is nested. Similar to the cognitive dimension of diffuseness, in a simple or small design, both the plain whiteboard and Calico are equal in that all parts of the design are visible and there may be little need to reference outside information. However, in larger projects, Calico may have more hard mental operations, because the design may be diffused across several canvases requiring the user to reference sketches outside the immediate space. Users may need to use
the intentional interface feature to move between canvases to reference other sketches, or use the palette to move content so that it can be referenced in the active canvas.

Calico further reduces hard mental operations in other situations. It allows users to both import content and copy content to the canvas they are currently in. First, image scraps allow users to import existing artifacts into a canvas to refer to while sketching. For example, a developer may import source they are working on, or a diagram created in another tool. Second, scraps allow users to move content so that it is positioned side-by-side to what they are currently working on. Third, the palette allows users to copy content between canvases so that they can reference old content in the new space. Fourth, the navigation button in intentional interfaces allows users to return back to the most recently visited canvas. While this does not negate the need to refer to parts of the design outside of the immediately visible area, the feature reduces the effort needed to do so.

7.2.7 Hidden dependencies

Hidden dependencies refers to representations that are dependent on one another, but the dependencies are not visible, because they are either not declared or not in a person’s field-of-view. As observed in past studies [22], sketches that cross several spaces result in having dependencies on other spaces that are not explicitly stated. Sketches that begin in one canvas in Calico may extend onto another canvas, or across mediums, such as their own computer or paper, if the users are using Calico as a complementary tool. This could occur, for example, when canvases depict different perspectives or related abstractions.

The intentional interfaces feature in Calico helps in declaring and finding these hidden dependencies on other sketches by providing features that group canvases of related content. Clusters in intentional interfaces help group canvases into sets of similar topics. Linking canvases into chains provides a second level of grouping that helps designers in finding related
sketches. These features do not solve the issue of revealing all hidden dependencies, but they do improve working with them by making those dependencies easier to find.

### 7.2.8 Premature commitment

A notation or representation that causes a user to *prematurely commit* means that it requires a person to make a decision before they have the information they need [64]. Sketches and content produced in Calico are typically used to support conversations or thinking through a design, and thus there is less of a concern to prematurely commit, because all content is provisional. The sketchy appearance of content within Calico visually re-enforces the notion that content is provisional. Also, the ease of changing content, such as drawing an “X” symbol over rejected content, allows for decisions to be rejected without permanently discarding the idea. In this regard, Calico is equivalent to the whiteboard, which also has these qualities of provisionality.

However, Calico further reduces premature commitment in comparison to the whiteboard by making sketches more easily copied and reproduced. Features such as scraps, the palette, and intentional interfaces make it easier to explore alternatives. For example, while on a whiteboard, a user may be hesitant to erase a large figure or deviate, because they do not want to lose previous work. In Calico, they can simply create a copy of it in a new canvas, or continue their work in a new blank canvas.

### 7.2.9 Progressive evaluation

Progressive evaluation refers to the ability to run a simulation or obtain feedback on a representation that is only partially complete in order to determine its correctness. While Calico does not directly support progressive evaluation, it does support users in manually judging
the “rightness” of a sketch by supporting the reviewing and explanation of sketches. Much like the whiteboard, designs may be manually reviewed and discussed by close inspection.

With large designs, Calico provides some additional benefit with intentional interfaces, which aids users in navigating between canvases to review them. However, as the sketch grows too large, it becomes progressively more difficult to manually evaluate all sketches.

Further, the fading highlighter increases a group’s facility review and discuss their content. By not leaving marks behind, the fading highlighter keeps the sketches cleaner. On a regular whiteboard, the designer is limited in the amount of marks they make over a sketch before the annotating marks make the original sketch unintelligible. However, this could be considered a loss in comparison to the whiteboard as well, as designers could potentially refer back to these marks to remind them of their evaluation.

7.2.10 Provisionality

Provisionality refers to the quality that indecision or options can be expressed [64]. Given that sketching is the primary mode of operation in Calico, individuals are free to break away from formal notation to declare multiple alternatives. Users can always erase a sketch to change a value, or simply cross out a sketch so that a design can be rejected without visually removing it from the design space. Calico goes further in supporting provisionality by enabling the quick generation of alternatives using duplication features such as scrap copying, canvas copying, and the palette. With intentional interfaces, users can explicitly express entire canvases as being an alternative by tagging it within intentional interfaces.
7.2.11 Role expressiveness

Role expressiveness refers to a reader’s ability to see how the parts fit into the whole design, and how those parts relate to one another [64]. In other words, a reader can independently interpret what each part in a sketch does without further information. Given that users have the flexibility to sketch any notation they need, the limiting factor in both the regular whiteboard and Calico is the space available to create representations. With enough space, the role of each part can be better understood if all details are visible. On the whiteboard, this means that a sketch must be sufficiently small. In Calico, this means that there is a virtually unlimited amount of space, but the user must then contend with a high amount of diffusion and low visibility. Intentional interfaces provides the means to manage the space needed to explore a design across multiple sketches to depict and especially name all parts of a design.

7.2.12 Secondary notation

Secondary notation refers to the use of formalisms that deviate from primary established notations. On both the plain whiteboard and Calico, users have the flexibility to overlay additional detail in sketches and use improvised notations as needed. The user is potentially only limited by the space they have available to express their secondary notations.

In Calico, however, secondary notations further benefit from the usage of scraps in that they stay with a scrap as it is moved and manipulated. Further, users may choose to represent concepts using scraps with connectors as an alternative to basic sketching, in which case they can use the shape of scraps, as well as the colors of connectors to overlay meaning. Users may add annotations to scraps, such as drawing a symbol in the corner of a scrap, text content in the scrap with a color that has an assigned significance, or by using connectors in creative ways.
While scraps help with secondary notations drawn inside a scrap, secondary notations made outside of a scrap do not move with a scrap. The negative consequence of this is that the annotation will become disconnected from the scrap once one of them is moved.

### 7.2.13 Viscosity

Viscosity refers to a medium’s resistance to change. Similar to a regular whiteboard, the contents in Calico exhibit a low viscosity because content can always be erased and redrawn in any manner.

Calico has two features which potentially reduce viscosity in comparison to plain sketching: (1) content is moveable, and (2) gestures make manipulating content faster. First, all sketched content can be moved using scraps, as opposed to the whiteboard where it must be redrawn. The capacity to move content means that sketched content is not “locked in” like a whiteboard, but instead content is less viscous, because it can be adjusted. Second, with sufficient familiarity with scrap gestures in Calico, moving content can potentially be done quickly with minimal effort. A user may be less likely to modify content if accomplishing that task requires a great deal of effort. In Calico, users do not need to switch modes or move their hand away from sketches, allowing them to move content fluidly without distracting them away from plain sketching.

### 7.2.14 Visibility

Visibility refers to how easily all parts of a design are visible, or at least how easy it is to juxtapose two parts of a design [64]. As with a whiteboard, a design is highly visible so long as it is small enough to fit within a single space. In Calico, designs that are diffused across multiple canvases will have much less visibility than designs that are limited to a
single canvas. The intentional interfaces feature attempts to address this issue by making it easier to navigate between canvases, and also to zoom out to the cluster view and view all canvases at once, albeit at a low-detailed perspective.

7.2.15 Summary

Overall, the majors factors that increase facility in Calico over the whiteboard are that intentional interfaces allows more content to be preserved whereas on the whiteboard it would be erased, and the fluidity of manipulation offered by scraps. They affect the affordances to the user by making content less viscous, more provisional, promote secondary notations, increase role expressiveness, and reduce error proneness. What is lost in these features is that designs may be more diffuse, and because of that diffuseness, require more hard mental operations, have less visibility, more hidden dependencies, and be harder to progressively evaluate. For each benefit gained through features, Calico loses the simplicity of the whiteboard, but attempts to mitigate these losses through the other features.

7.3 Overall strengths and weaknesses

Having examined the features from a practical perspective in Section 7.1 and at a theoretical perspective in Section 7.2, this section examines the overall strengths and weaknesses of Calico as a whole.

7.3.1 Strengths

Strength: It supported all design behaviors
Cumulatively across all groups, individuals performed every design behavior, and performed each at least once using Calico’s features. While not all groups performed every design behavior using the advanced features (for example only the research group used intentional interfaces to retreat to previous designs), each group made major use of the features to accomplish actual work. Existing tools (presented in Chapter 8) support a small subset of the design behaviors, locking users into features that support two to five design behaviors at the expense of support for other design behaviors. On the contrary, with Calico, users used its features to, among others, exercise design behaviors as diverse as switching notations at will, juxtaposing high fidelity images against low quality sketches, mentally simulating over a sketch with other designers, or working asynchronously and immediately break apart to work in their own canvas. That Calico was flexible enough to support all design behaviors without major interruptions or requiring another tool was a strong point.

Strength: It provided support when they needed it, as opposed to forcing it upon them

While all features were used to support their respective design behaviors, the designers only used those features when it benefited them. They, for instance, did not need to always create all sketches as scraps, they could instead transform sketches into scraps when they were ready. They also did not always need to organize their canvases using intentional interfaces, they could simply press “new canvas” or “copy canvas”, and organize their canvases at a later point. This behavior is in contrast to other tools such as the UML editor Rationale Rose or a wire framing tool Balsamiq, which may cause the user overhead in expressing their ideas in terms of the tool, rather than what is natural to them. A strength of Calico is that users could simply sketch and ignore the advanced functionality as long as they did not need it.

Designers will not use the advanced features when they do not need them (as evidenced in Chapter 6). Regardless how minimal the effort may be, each feature brings with it some “cost” that the user must incur in order to gain the benefit of that feature. Scraps enable
rapid interaction with sketched content, but require the user to think in terms of objects and manually group them. Intentional interfaces enable the user to have more free space, but requires the user to think in terms of multiple canvases with low visibility. A user may not need the benefits offered by these features and may simply want to sketch without structure imposed on their work. A strength of Calico is that it allowed them to do that, but, when ready, to also convert regular sketches into scraps using the press-and-hold gesture, or partition the contents of a single canvas across several canvases by pressing the “copy canvas” button and organize them using intentional interfaces.

**Strength: It supports a wide range of design activities**

All groups engaged in different forms of design, yet each found Calico useful in their own work. The OSS group engaged in late-stage maintenance design, building new components on top of a mature open source system and refactoring source code in a co-located setting. The interaction designers performed early phase design, which involved requirements gathering by processing interviews to build personas. The research group was engaged in implementation and user interface design, building and refactoring the first version of a software tool in a globally distributed team.

Each group created their own culture in using the tool and performed certain design behaviors more than others. The OSS group created narratives, depicted across multiple canvases, explaining how data is moved through their system, and, to do so, extensively used the fading highlighter to mentally simulate and intentional interfaces to move between levels of abstraction and perspectives. The interaction design group invented their own notations using scraps, refined these representations, and generated dozens of different perspectives. The research group carried out their sessions distributedly, explaining designs to each other using Calico and Skype. They particularly used scraps extensively for building process flows.

Calico supported each of the groups over multiple days of work, which included brainstorm-
ing, sharing, refining, and archiving [81]. The groups adapted Calico to their existing workflow by sketching or importing diagrams, working, and later archiving their work by e-mailing or referring back to old sketches in Calico.

**Strength: It can create a variety of diagrams**

Across the design sessions, Calico showed that it could be used to create a variety of different types of diagrams. Some diagrams, such as box-and-arrow diagrams, benefit greatly from scraps because they could more easily be moved and related. Other types of sketches, such as the image scraps placed along one and two dimensional plots benefitted from being able to tag the elements with colors and move them along the plot to organize them. For complex diagrams, users can use plain sketching, but with scraps, they have a general purpose tool to group parts of these sketches and juxtapose the ideas captured by them by moving the scraps next to one another.

**Strength: Presents a minimal hurdle for first use**

A strength in Calico’s core design is that users can use its most basic features without training. New users unfamiliar with Calico simply approached the board and dragged their finger to draw without an introduction to the tool. The minimal overhead allows new users to begin working without the overhead of creating a new project, reserving a space, or investing themselves in extensive training. This is helpful when sharing or explaining a design to new users, who can begin participating in a design session without any training.

**Strength: It’s interaction is fluid**

Calico uses a minimal set of modes for manipulating content, which leads to a more fluid experience than traditional digital sketching tools. Plain sketching is fluid because the designer is interrupted, at most, to reach for a new pen, which allows them to focus on the content of their sketch rather than the tool they are using to make it. The combination of
scrap gestures for creating scraps and the bubble menu fit well with the fluidity afforded by a regular whiteboard. Manipulating content using Calico requires fewer steps than other digital sketching tools to manipulating content, such as Adobe Photoshop.

**Strength: The basic interaction mechanisms are strong**

Overall, Calico contains a strong basic set of interactions. The groups did not find Calico lacking in basic features. Despite Calico not having advanced drawing features such as shadows, blur, etc., or having advanced modeling features such as different types of arrows, shapes of boxes, etc., users were clearly able to accomplish their work. Users valued the basic set of features in Calico, such as the eraser, undo/redo, colors, stroke thickness, and the set of Calico’s advanced features. They found Calico’s features sufficient to carry out their sessions, and found Calico’s features natural to use after a period of time, in particular scrap gestures and the bubble menu. The same could not be said for the interaction scheme of Calico Version One, which users found confusing.

**Strength: It allows for creative uses that were not expected**

Users adapted the mechanics of scraps to invent uses that went beyond what they were trained to do in the initial tutorials. The use of Calico by a member of the OSS group to refactor their source code was a deviation from what was expected. Calico was envisioned to support the sketching and manipulation of abstractions, but the OSS group member found the writing on the electronic whiteboard good enough to write pseudo code and found the use of colors helpful as a secondary notation. The use of colors to tag scraps in both the interaction design group and research also came as a surprise. The use of Calico by the research group to keep track of what they had and had not implemented further corroborates Newell’s notion that sketching serves as an external memory [53], but this particular use also was not anticipated.

**Strength: Free space is always available**
Users explored new ideas in Calico without erasing old ones. As mentioned in the previous section on cognitive dimensions, this encourages role expressiveness by allowing users to fully express their designs, but brings with it the problem of diffused diagrams that cause low visibility, hard mental operations, hidden dependencies, etc. While these negative consequences are concerns that should be addressed further, the value provided by easily attainable free space was an overall positive outcome for groups, particularly because all of the groups created much more content than would fit even on a handful of traditional whiteboards.

**Strength: Linking canvases creates narratives that are more easily understood**

The OSS group and one member from the research group demonstrated that intentional interfaces is useful for creating narratives from sequences of canvases. Similar to how designers may refine sketches by converting lists into UML class diagrams, members from the two aforementioned groups manually reorganized the order of their canvases using canvas links in order to create a story from their sketches. These groups later shared their sketches, and compared the act of stepping through these canvases to giving a Powerpoint presentation.

**Strength: Sketched content is always available**

A quality of Calico that members from the OSS group and the research group valued was always being able to access their canvases. Group members may not always have access to the physical equipment, and being able to access the content on their own computers was helpful. Indeed, some members took to creating content beforehand, to then show during a design session. Clearly, the distributed nature of Calcio was important.

**Strength: The system is distributed**

A strength of Calico was that it enabled groups to not be locked in to a single device or board. For the OSS group, this led to a more informal setting in which members spontaneously broke into small groups in meetings by handing tablets back-and-forth, sketching over the
diagrams, and having their annotations displayed on the large electronic whiteboard. With multiple tablets, multiple team members could talk simultaneously without a single arbiter at the whiteboard blocking content, which is a known issue in group settings at a regular whiteboard [72]. For the globally distributed research group, the consequence was that remote members could be active members in meetings using Skype, by sketching their ideas, rather than remaining a passive contributor by voicing their opinion.

Strength: Users are not locked in to synchronous work

A person at the whiteboard that wants to write something, but cannot, may “spin their wheels” and ignore conversations until they can externalize their thoughts [58]. A strength of Calico is that users can branch off into their own canvas, either by creating a new canvas or copying the one that is in active use, and explore their own ideas independently immediately when they needed to. In meetings of large groups, this frees members to generate their own alternatives when inspiration strikes them, and later request everyone to join their canvas so that the designer can explain their independent work.

7.3.2 Weaknesses

Weakness: Juxtaposing sketches on different canvases is difficult

A weakness in Calico is that sketches across canvases are difficult to juxtapose. Feedback from users indicated that they wished to refer to content from other canvases. In order to refer to past sketches, users either used the palette to copy sketches so that they are immediately available for reference, or they used the navigation buttons to rapidly jump between previous canvases and the canvas in which they are currently working. These features were used to compensate for the inability to place sketches side-by-side.

Weakness: Users that do not invest in organizing canvases will experience diffi-
In previous work, there is much evidence that the Calico grid helps in design. It provided an intuitive interface to organize and partition work between canvases. We believe that intentional interfaces is a step forward from the grid, because it adds support for organizing and especially relating canvases in more flexible ways. However, this feature has not had the opportunity to be polished and iterated on to improve its usability. One weakness in the current version of intentional interfaces is that users that do not actively manage their canvases will likely have difficulty in knowing where their sketches will be located in the cluster view. Users that manually manage their canvases do so by linking their canvases using tags. Users that do not manage their canvases create radial clusters of canvases, each without a name, which makes it difficult to find them again. Future iterations of intentional interfaces in Calico will need to address this issue.

**Weakness: Reviewing a large number of canvases is tedious**

Moving across several canvases using the cluster view is a slow experience that can become tedious. A consequence of creating diffuse diagrams with many canvases is that the cluster view will need to adjust the size of canvases in order to display all canvases. The resulting effect is the canvases become too small to distinguish when there are a large number of them. Other features help to mitigate this problem, such as using the navigation buttons, or using the breadcrumb bar. However, members in the OSS group reported that, despite the slow interaction, they prefered to use the cluster view with previews to review sets of related canvases. Doing so, however, is far from a completely fluid act.

**Weakness: Does not follow traditional interaction patterns, requires users to learn new gestures to manipulate objects**

While the basic sketching functionality of Calico can be used by new users without training, the advanced feature set of Calico may take new users some time and practice to become
proficient. A further issue is that users may approach Calico with a pre-conceived set of affordances that is based on their experience with other touch based devices. For example, members of the OSS group expected the large electronic whiteboard to use the same gestures as their mobile phone, while the interaction designers expected it to behave like Microsoft OneNote. Calico introduced another set of interactions that the users needed to learn.

**Weakness: It stops short of something that becomes an actual notation**

While Calico supports users in creating approximations of formal notations such as UML, it stops short in delivering the behavior and expressive power of a formal modeling tool. A strength of Calico is that it supports working with informal notations and refining notations into more notationally complete approximations. However, a designer may reach a point in which they are ready to add additional notational detail to their diagrams, such as double headed arrows, cardinality, and other such embellishments. While Calico does not currently provide support for these elements, such support has been observed as being needed in late-stage design, after a designer has committed to their design and begins recording lower-level, detailed design decisions.

**Weakness: The setup is not lightweight**

A problem observed in the setup of Calico was that, in order to use Calico, the system needed to be prepared and maintained by an administrator. While this is not specifically a weakness of Calico itself, it is highly relevant to Calico because, based on my observations, individuals will not use Calico if there is a delay in using the system. Instead, they will turn to using a normal whiteboard or a pad of paper.

In practice, the large electronic whiteboard needed a dedicated system that was always running so that each group could begin using Calico simply by turning on the attached projector. In order to launch Calico on a personal tablet, the tablet required the installation of the Java Virtual Environment, which at times delayed meetings when new members needed
to install their software. However, once group members performed the initial setup of Calico on their own machine, connecting to the Calico server became a more lightweight action.

7.4 Interviews with users

In the previous sections, I examined Calico and its features from a number of different perspectives. Here, I add the final perspective: Calico as seen through the eyes of the users from the three field studies. Specifically, I interviewed each group about how they used Calico in their own work, and how it changed their design practices.

The interviews confirmed the observation of each group using Calico very differently. Each group was engaged at a different stage of the design process, or a different project, with different amounts of people, and in different settings. The groups, not surprisingly, therefore reported different strengths and weaknesses in Calico.

The interaction designers used Calico to engage in activities they would normally perform with physical artifacts on a physical whiteboard or table. As such, many of the comments that came from the interaction design group related to Calico’s strengths and weaknesses as they compared to, for instance, the printed photographs and sticky notes that the interaction designers normally would use. For example, a strength that the interaction designers found was its ability to import images as image scraps and flexibly arrange them. Explaining why this was important, one member of the interaction design group stated,

“It’s really hard to say ‘that person or this person’. Much easier to speak to a face.”

They particularly found that Calico’s strengths were those that enabled actions not easily done with physical artifacts. For example, replication of artifacts encouraged the interaction designers to explore more perspectives than they would have, had they used actual physical
artifacts. Calico’s features further allowed them not only to arrange the images themselves, but also the content around those images. For instance they brainstormed text scraps and grouped them, drew axes, and informally clustered text and image scraps along the axes.

However, they also reported some experiences not on par with what they were used to and expecting. They viewed the gesture of moving scraps (click and hold) too slow, with one member of the interaction group stating that:

“the overhead in manipulating [scraps] was too much.”

Further, they found the space available in canvases being too small compared to the boards they normally use, stating that:

“[we] [were] blocked by the physical limitations of the [electronic] board.”

Despite these obstacles, they did make dozens of canvases full of sketches and scraps. They made real progress in a real design task within their professional work.

The OSS group made use of Calico in group design discussions of three or more people, and also personal sessions. To them, a strength of Calico was that it increased their ability to produce more designs in their meeting space. For example, intentional interfaces encouraged members to brainstorm more, with one member of the group stating:

“we’re more willing to draw any random thing on there because we know that we can erase it, or go to a new canvas at any point. I would say more random ideas get thrown on there.”

They used Calico for both meetings and personal design sessions. When working individually, one member of the group turned to Calico in order to gather their thoughts. He states:
“I [had moments] where I needed to get my thoughts written out as far as [what] I was trying to design.”

The OSS group in particular took advantage of free space to create dozens of canvases. For example, they consecutively chained canvases through the intentional interface feature to generate sets of canvases that together formed narratives, such as how data steps through software components, or how a user interface behaves.

The members of the group worked in a shared space, and used Calico to overcome the issues they had previously, such as the unavailability of whiteboards. A member from the OSS group states:

“The fact that someone can work with their own tablet or computer, like [a member of the team] did with his alternative view, is something really powerful to do. If you try to do that in real life... we’re blessed with two whiteboards, but some companies may only have one. It makes it harder to do that. Especially when someone is already at the whiteboard discussing something, and you want to bring in an alternative perspective but you need to wait until they’re done doing whatever they’re doing. That definitely makes groupwork easier.”

In meetings, they used both the tablets and the electronic whiteboard to work concurrently in Calico. One member of the group states:

“At one point I grabbed a tablet and started drawing..., I got a new canvas while everyone was discussing one of the previous canvases. I drew this new [design], and, when I was done, I told everyone to go to this slide, and we started talking about [it]. It’s nice to be able to just switch to something right there in front of you.... we could do the same thing with two whiteboards, but then we would need to move everyone around [the room]. Being able to work simultaneously is nice.”
Some of their use was also unexpected. For example, the sessions involving source code came as a surprise as the action of copying and pasting screenshots of actual code into Calico was not part of the design behaviors. These sessions became some of the most interesting, as they involved a great deal of impromptu notations to step through source code. The member of the OSS group that performed this action found it helpful in their design, stating:

“It was really useful to switch [to Calico]... I used a lot of color. It ... helped me identify... which kinds of objects were important. The other thing was that I wanted to look at the XML code that was relevant to this diagram, so I went on my machine, logged in over the web browser, took a screenshot of my code, and pasted it into my canvas. That way, when I came to the board, I could look at the code from the screenshot, and do my work based on that. So that was very helpful.”

The research group presented a third unique setting as it involved meetings that were distributed, one member who did a lot of design up front before the group design sessions began, and the creation of very large process diagrams that described the entire behavior of their software system. As opposed to the other groups, the research group members were more isolated from one another, and used meetings to review their work, and plan what to do next. The research group viewed one of Calico’s strengths as being able to carry on meetings with remote members, with one member stating:

“I think the biggest benefit was crossing space and time. Being able to pause and resume...the main benefit was that, after I left, we were able to reference all those things that we created while I was there pretty easily.”

They viewed Calico as more beneficial than photos of a whiteboard, stating that

“you can’t go back and edit that.”
Further, one member of the research group prepared his work ahead of time for meetings, stating:

“I could have done this in a PowerPoint as well, [but] it’s much better [to have done it in Calico] because later when I show it, I can have people changing it, and in PowerPoint you cannot sketch and draw.”

This particular member further found the visual structure of intentional interfaces helpful when reviewing and returning to work from past meetings, stating:

“designs get very complex... you want to keep a history of what you’ve done, the branches that you’ve pruned. Having a structure is essential. If you’re designing a complex thing with stages and you’re trying to tell a story, you can say: okay we’ve tried that, would you like to see all this path we went through? If you don’t have the structure you’ll have to create it somewhere else. [You save time] if it’s already here…”

During their intensive, week-long design session, they used Calico to capture and modify the design of their system. Typically only one member of the group used Calico to record the group’s design decisions while the group as a whole worked from their laptops. One member states:

“Somebody was talking, and another person would draw the connectors on the board. So sometimes we would say, this is the label of the next thing, and sometimes one person would write down the label [into Calico] while the next person is talking about it. ”

Calico served as a virtual meeting space for the members that helped maintain continuity of meetings across the duration of the project.
The takeaway is that each setting introduced a unique way of using Calico. Each group had a different set of needs, in a different setting, and, yet, Calico held up to a significant portion of their needs. Each group followed through with their design sessions and completed work in actual projects without abandoning their use of the tool. That is not to say that it was a perfect match for all groups. One of the teams, the interaction design group, did say that they would not continue to use Calico because its features did not match their needs, such as an infinite canvas and the ability to support a greater number of images on a single canvas without slowing down. This may either mean that their needs would be better suited by another tool, or further research is needed with respect to how Calico could be adjusted to suit them better.

Further, from the interviews, three observations stood out: (1) Calico did not interfere with the normal design process of two of the three groups (the OSS and research groups), and, despite difficulties, the interaction design group performed non-trivial design sessions, (2) Calico deployments still have issues that need to be addressed that are intrusive, and (3) Calico led to positive changes in their design habits.

First, the OSS group and the research group reported that Calico and its features did not prevent them from carrying out their design as they normally would have, while the interaction design group reported that Calico somewhat got in the way of their normal design activity. The OSS group reported that they did not feel any loss of expressive control in using Calico in comparison to the whiteboard, and reported that they normally would have performed many of the same activities on the whiteboards available in the immediate area. The research group reported that they formerly carried out their designs on the whiteboard, and transitioned to using Calico for the same activities. The interaction design group reported that the kinds of sketches they created in Calico closely approximated the kinds of sketches they would have created on the whiteboard, but they found that the quality of sketch input forced them to write large and that the system lagged at times, particularly
when using more than thirty images in one canvas, which they found to be distracting during their sessions.

Second, a set of issues that negatively impacted the design experience arose among all three groups. Nearly all groups reported that the large electronic whiteboards diminished the quality of their own handwriting, forcing them to either write slower, or write larger. The OSS group connected to their Calico server using either tablets or their own computers, which they reported was preferred in producing clearer handwriting due to precise input. All groups used text scraps to produce legible text as well. All groups also reported that launching Calico for the first time on their own machines was sometimes a hurdle. The OSS group reported that it was less of an obstacle for them, because of the availability of tablets preloaded with Calico. The research group reported trouble setting up Calico on the laptops of new members of their group.

Third, all groups reported some positive changes to their existing design habits. All groups reported that they had a sensation of having more free space, and reported that they created more sketches than they normally would have as a result. The research group reported that they created more complex sketches as a result of scraps and connectors, which helped them address a deeper level of complexity in their design. Both the OSS group and the research group reported that, since they did not feel a need to delete unused sketches, they returned to old sketches more often.

7.5 Summary

Having presented my analysis of the experiences reported in Chapter 6, I now return to the research question introduced in Chapter 3, what minimally invasive, coherent set of features can be designed that is sufficient to effectively support these design behaviors?
Addressing the first part of the research question, the observations made thus far show that Calico’s features were indeed minimally invasive in real world design activities. Minor issues did exist that users found invasive, for example limited space in a canvas, poor quality of handwriting on the large electronic whiteboards, and, in the case of the interaction designers, degraded system performance when using more than thirty image scraps in a single canvas. Despite these issues, all users were able to use Calico to perform actual work in real design sessions, and users saw Calico as improving their capability to design in comparison to the regular whiteboard.

Addressing the second part of the research question, the features do coherently work together. Table 7.3 succinctly summarizes the contributions of each feature to the set of fourteen design behaviors. Each feature supports multiple design behaviors, and they clearly are integrated, one building on top of another. Scraps and the palette are very much integrated, where both work together in design behaviors 2, 4, 6, and 14. Scraps were also integrated with intentional interfaces, both supporting design behaviors 5, 6, 7, and 9. The fading highlighter, however, stands apart from the other features, supporting design behaviors 8 and 13. While it stands alone, it is powerful in what it does, and does so in a minimal and lightweight fashion.

Table 7.3: The set of design behaviors and the features that supported them.

<table>
<thead>
<tr>
<th>Design Behavior</th>
<th>Scraps</th>
<th>Palette</th>
<th>Intentional Interfaces</th>
<th>Fading highlighter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. They draw different kinds of diagrams</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. They produce sketches that draw what they need, and no more</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### Table 7.3 — Continued from previous page

<table>
<thead>
<tr>
<th>Design Behavior</th>
<th>Scraps</th>
<th>Palette</th>
<th>Intentional Interfaces</th>
<th>Fading highlighter</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. They refine and evolve their sketches over time</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. They use impromptu notations</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. They move from one perspective to another</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>6. They move from one alternative to another</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>7. They move between levels of abstraction</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>8. They perform mental simulations</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>9. They juxtapose sketches</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>10. They review their progress</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>11. They retreat to previous ideas</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>12. They switch between synchronous and asynchronous work</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>13. They explain their sketches to each other</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

*Continued on next page*
Addressing the third part of the research question, the set of four features did indeed support all design behaviors. Scraps and the palette supported designers in the kinds of sketches they created (design behaviors 1 – 4). They allowed diagrams to “grow organically”, have rich conversations over the runtime behavior of components, and build relationships over unrelated sets of data. Intentional interfaces, scraps, the palette, and the fading highlighter helped support the way designers navigated between sketches (design behaviors 5 – 11). Intentional interfaces helped designers find related perspectives, alternatives, and levels of abstraction, and did so by attaching related canvases with tagged links and providing an order to them. This also helped designers in reviewing their progress and retreat to previous ideas. The fading highlighter further supported designers in mentally simulating their designs by allowing them to write on a sketch without leaving marks. Lastly, intentional interfaces, the fading highlighter, scraps, and the palette helped designers in collaborating with one another (design behaviors 12 – 14). Intentional interfaces helped designers in moving between synchronous work and opportunistically branching off to asynchronous work. The fading highlighter enabled designers to explain their sketches to one another from remote computers. The palette further allowed designers to bring work together from different canvases, and scraps also helped in bringing work together.
Chapter 8

Related Work

Existing sketch tools can be classified into two categories: (1) those that interpret sketches with the purpose of turning them into representations that adhere to certain notational conventions, and (2) those that support sketching activities in general. Johnson et. al. [39] provide an extensive review of all of the sketch tools contributed until 2008. Here we make a selection of the tools and approaches most relevant to Calico, and also include more recent work.

8.1 Tools that interpret

The tools within this category focus on assisting designers by transitioning their sketch into representations that adhere to certain notational conventions. For instance, sketched boxes and lines between boxes may be turned into UML diagrams, or informally drawn user interface widgets into an interactive user interface prototype. While we purposefully chose not to interpret sketches in Calico, these tools remain relevant as background work.

Many tools that interpret sketches focus on generating prototypes for user interfaces. One
such tool is SILK [44], which allows users to generate a GUI prototype from a sketch that can be used by its designers to walk through how the UI would work. Once they are satisfied, they can even generate the corresponding Java code that implements the GUI. SILK was the first tool of its kind, and stressed both the importance of working in a low-fidelity environment, and the value of rapid iterative design work. SILK differs greatly from Calico in focus. It is limited to UI design, and focuses on only a few design behaviors, most importantly mental simulation and refinement.

Another tool, DENIM [54], allows users to create low-fidelity mockups of web sites that the user, again, can walk through to get a sense of the flow of the web site. DENIM builds on the concepts of SILK by both supporting interpretation of sketched content and navigation between web pages (as opposed to desktop user interfaces). DENIM further contributes the notion of multiple levels of hierarchy for navigating design sketches, in which the user zooms out to navigate the network of web sites, and zooms in to view sketched content on individual web pages. DENIM’s use of multiple levels of levels of abstraction for navigating web pages, and linking web pages, is similar to Calico’s intentional interfaces, but is highly specialized for designing web pages. Further, DENIM still only supports a handful of design behaviors (i.e., refinement, impromptu notations, and moving between levels of abstraction).

Other tools target multiple domains, such as InkKit [66] and Freeform [16], which created usable prototypes from sketches of user interfaces, lists, ER diagrams, etc. The design of these two tools stresses the importance of retaining the sketchy look-and-feel of what is drawn, which is based on the insight that visually beautifying content too early can be harmful to the design process [73]. Unlike previous tools, InkKit and Freeform also support different types of sketches, such as user interface and ER diagrams. InkKit specifically uses different windows within a space, each that is configured to interpret one pre-defined notation. Sketches in one window are explicitly linked into another window, which may be used to create a storyboard, or to populate a widget (such as a list populating a dropdown...
Sketches are interpreted to generate a working prototype system that, once again, can be walked through. By interpreting sketches from multiple perspectives, these systems provide rich prototypes for feedback at a low cost of effort for the designer. These systems are similar to Calico in that users can sketch a system from multiple perspectives, however these tools restrict users to using a pre-determined set of notations. Moreover, from a design perspective, these tools only support a subset of design behaviors, which include using different kinds of representations, refinement, mentally simulating over sketches and moving between different perspectives – all are limited to the set of possible interpreted notations.

Another group of interpretive sketch systems includes those that recognize UML. The earliest such tool was Knight [19], which allows designers to create partial UML diagrams by selecting a portion or all of their sketch to be interpreted as UML diagrammatic elements. Knight was inspired by studying software designers at the whiteboard, and contributes the observation that software sketches begin vague, but become refined over time. The Knight tool allows designers to sketch and convert their sketches into a beautified UML representation by manually triggering the recognition process. Both Calico and Knight allow users to refine their drawings from plain whiteboard sketches into objects at a later point, however Knight automatically interprets UML notations for the user and offers advanced notational support, while in Calico users must manually refine sketches and are limited to the support by provided by scraps. Knight, as the other tools discussed thus far, supports only a subset of the design behaviors, namely refinement and moving between a limited set of perspectives.

The tool SUMLOW [14] further advances on the work of Knight by allowing interpreted sketches to retain their sketchy appearance after interpretation (citing InkKit as inspiration), and allows the mixing of different UML diagrams (e.g., class diagram, sequence diagram, use case diagram). While both Knight and SUMLOW make UML diagrams a less viscous representation by allowing users to simply sketch, the retainment of the sketchy appearance after
interpretation introduces a level of provisionality that was not available in previous sketch interpretation tools. Both Calico and SUMLOW are similar in this philosophy. Hosking and Grundy later integrated SUMLOW into a full software design environment, Marama-Sketch [32], which designers can use to sketch diagrams in a broad variety of notations. Marama-Sketch brought tighter integration between sketched models and actual code, a quality which Calico itself does not have. At the same time, Calico still supports many additional design behaviors that SUMLOW and Marama-Sketch do not, notably moving between alternatives, levels of abstraction, mental simulation, and collaborative behaviors.

Numerous other UML-oriented sketch tools have been created, generally following similar strategies (for a survey, see [39]).

Another approach to sketch recognition is to build a general purpose framework for interpreting sketches. LADDER is such a tool [36]. It provides an extensible, general purpose language which can be adapted to many notations, including software design notations, biology and chemistry notations, and military symbols. Of the many interpretive frameworks (e.g., SATIN [38], LADDER [36], InkKit [66], SketchML [2]), most target a single, or small set of notations. Based on the observation that many frameworks use different engines, yet that each still suffers from mixed success in automatically recognizing sketched content, Dixon provides an approach that queries multiple engines simultaneously, and ranks the results by their probability of success [24]. These tools vary greatly from Calico, given that they focus on sketch recognition, while Calico explicitly chooses not to recognize sketches.

Other tools allow sketches of mixed notations to coexist in the same area as well, though interpretation is left to an explicit manual choice by the user rather than automated recognition. An example of such a system is Patches, in which the user can create sets of irregular shapes called “patches” [43]. These can be moved around, be made translucent, and, most importantly, a patch can be assigned an interpretation that designates it to become a list, outline, table, or other element. The functionality, as well as look-and-feel, of the scraps
functionality in Calico is similar to the Patches system. However, they differ in their goal, where Patches views its functionality as providing lenses over existing sketches, and uses patches to offer transparent layers on top of each other. Calico, on the other hand, uses scraps as building blocks to create more complex representations. More so, Calico’s primary aim is to support a wide range of design behaviors, while Patches supports just a limited set (i.e., different kinds of diagrams, refinement, impromptu notations, and moving between perspectives). Patches inspired the overall functionality of the sketch framework SATIN [38], which was later used to create the tools DENIM (previously mentioned) and Designer’s Outpost ¹.

8.2 Systems that support sketching activities

The second type of sketch system is aimed at supporting the general design activity. These systems help organize sketched artifacts during meetings, make them retrievable, and help manage and maintain the plethora of sketched design content that gets generated during design sessions.

Colab, created for the Liveboard system at PARC, was the first system to work with this hardware, and acted as a meeting tool [75]. It had a simple, uncluttered interface with a mode selector and a visible set of saved sessions in the fashion of a filmstrip. The next system developed for the Liveboard was Tivoli [62], which used the filmstrip metaphor to manage sketches in a side panel. Tivoli also organized sketches into chunked items for easy moving, and had additional functionality for creating empty space. Tivoli did not attempt to recognize any of the drawn elements, but rather provided intelligent support for common tasks, such as creating lists and checking off items in lists. After Tivoli, Dolphin [77] introduced a client-server architecture for group collaboration and allowed users to create

¹ mentioned in the next section
links between canvases. Dolphin was the first whiteboard system to provide hyperlinks between content in canvases, which would not appear again in other tools until the creation of DENIM ten years later. These three systems show a remarkable foresight in their underlying ideas. Chunked items in Tivoli are a precursor to scraps, and hyperlinks in Dolphin operate similar to tags in Calico. Distributed operation also was supported. Compared to these systems, however, Calico has made significant strides in refining these early concepts, making them fluid in use, and enhancing them (for instance through intentional interfaces, which supports spatial organization of sketched content, as opposed to the temporal organization in the Liveboard tools). Moreover, Calico succeeds in bringing them together in a single system.

Many other tools provide methods for managing the presence of multiple sketches within a fixed space. PostBrainStorm [33] uses the border areas as zones where sketches auto-shrink, thereby minimizing space use by elements that are temporarily set aside. PostBrainstorm further allows users to place a physical picture on a whiteboard, and have the same picture automatically appear in the system. While Calico does not support such physical object scanning, the importing of existing photos showed itself to be an important feature in its use.

Flatland [51] automatically groups sketches into clusters, provides cluster-level functionality, and moves them out of the way when more space is needed based on where the user starts drawing. Flatland was not the first to provide clusters (e.g., Patches), but Flatland does make them the center focus, and offers rich support for organizing lists, creating maps, etc., as well as allowing users to move backwards and forwards in the history of clusters. Scraps, in comparison, do not support a specific type of representation, but offer the benefit of being stackable and relatable. Moreover, they form the basis for supporting numerous design behaviors, while Flatland only supports a few (i.e., different kinds of representations, sketching only as much as needed, and impromptu notations).
Range [41] extends the approach taken by Flatland by using a person’s physical proximity to the board as a trigger to create space. It also presents past sketches to the designer when they move physically away from the board in order to provide inspiration for creative work.

Overall, many of these systems attempt to support the designer at the whiteboard in general activities such as clustering objects and making space without the designer manually requesting this space. Calico does not provide such ubiquitous support in a single whiteboard, but instead allows users to manually organize their sketches using intentional interfaces. Such manual organizing of space has value in and of itselfs, as the act of organizing is a an important part of the design process [55].

A third set of tools specifically targets the organization of multiple sketches and canvases. Bellamy et al. [9] presents an ideation tool for building user interfaces. Users sketch the states of a user interface, which are presented back to the user as a set of storyboards. Additionally, the history of the user’s exploration is presented back to the user as a tree that includes undo actions. Other systems target the visualization of separate canvases, such as IdeaVis [28], which presents a visualization for a network of whiteboard canvases. The IdeaVis system captures hand-drawn sketches using a camera, and presents them back to a co-located sketch team using an “interactive hyperbolic tree visualization”, in which users can navigate a network of canvases in which the visual size of canvases becomes larger when in close proximity to the active canvas. Their studies showed that automatically making relevant canvases large helped users maintain focus to important sketches. This system is most similar to Calico’s intentional interfaces in that they both visualize a network of canvases, however intentional interfaces maintains a fixed size for canvases. At the same time, intentional interfaces supports types of relationships as well as the ability to manually organize the canvases. Calico, also, supports many additional design behaviors, such as different kinds of representations, refinement, mental simulations, and collaborating over sketches.
Another set of tools targets the collaboration aspect of electronic whiteboard systems. Designer’s Outpost [42] bridges the gap between physical Post-It notes and digital content, transitioning content of Post-It notes into digital artifacts that then can be organized in various ways. The core functionality of this system allows users to create links between digital Post-It notes, group Post-It notes into clusters, and zoom in and expand individual Post-It notes to work with them in great detail. Designer’s Outpost provides novel methods of interacting with remote users, such as showing a user’s shadow on the whiteboard in order to communicate gestures and body language. Calico, in comparison, does not communicate body language, but instead uses the fading highlighter to communicate transient information. Moreover, Calico’s aim is different. While the focus of a design session in Designer’s Outpost is on the hierarchy of Post-It notes, which leads to strong support for brainstorming ideas, Calico’s focus is on the software design process. It therefore provides support for design behaviors such as using different kinds of representations, refinement of sketches, and impromptu notations.

TEAM STORM [35] presents another approach to addressing the issue of public and private spaces. Users can create sketches on their own devices, and have the option of sharing their sketches in the public space, in which other users can interact with and provide feedback on sketches. Further building on the concept of a shared collaborative space, the GAMBIT [71] system addresses the issue of using multiple devices together, including large displays, laptops, tabletops, and phones. In studies of GAMBIT, researchers observed a trend in sessions in which users first build mental models of their domain, construct scenarios, and lastly sketch prototypes using GAMBIT. In comparison, sketches in Calico are always public, but users can move to a separate canvas to work asynchronously, thus supporting public-private work in a much more informal manner. Calico also provides support for the design behaviors pertaining to the kinds of sketches that designers make and how they navigate them.
Tele-board [34], is a distributed whiteboard and Post-It Note tool. Similar to Designer’s Outpost, it allows users to generate sticky notes from remote locations, group them, and review whiteboards in a history viewer. In a three month study at globally distributed offices, they found that this system was considered beneficial, though with the caveat that remote users found the system more beneficial than users in the company’s central office.

Tele-board and Calico share some similarities. They both support designers moving between perspectives, which Tele-board does using a slidesorter. Users in both can switching between synchronous and asynchronous distributed work by moving between canvases. However, Tele-board differs from Calico in that it does not support different kinds of representations.

Lastly, Camel [13] integrates a whiteboard system into Eclipse, allowing users to post UML diagrams onto poster boards. In doing so, Camel provides software teams value by supporting code reviews and design walkthroughs in an informal, sketchy manner: developers can simply write and draw the code being presented. Compared to Calico, Camel supports the design behaviors of using different kinds of representations, which can be copied from the formal representation or hand-drawn, designers may switch between synchronous and asynchronous work by switching between digital poster boards. It does not, however, support refinement of sketches.
Chapter 9

Conclusions and Future Work

Today’s distributed development practices have brought a renewed interest in being able to easily share information of all kinds. While video conferencing has been around for decades, more recently tools such as Google Docs, EverNote, and Dropbox have truly changed how people work. Omitted from these tools to date, however, has been the ability to share and work together on sketches – informal drawings that quickly capture and ground ideas. Particularly when it comes to distributed collaborative software design, sketching is crucial. Sketching improves design by allowing designers to quickly externalize their thoughts so that they can be viewed, analyzed, and discussed, and empowers them to express their ideas with a level of flexibility that is unmet by alternative styles of tools.

In this dissertation, I address this issue and make the following four contributions:

Contribution 1: The establishment of design behaviors as a necessary quality to support in informal design at the whiteboard.

Thus far, research has provided relatively abstract and typically high-level reasons of why software designers use the whiteboard, and how they use it. This dissertation extends that
work by providing a more concrete list of design behaviors that tool designers can use as a checklist for tools that aim to support informal software design. The design behaviors span three categories: (1) the kinds of things software designers sketch, (2) how software designers navigate between their sketches, and (3) how software designers collaborate over their sketches. Further, tool designers that wish to support early design work should keep in mind that software designers perform these design behaviors in rapid succession, and tools that limit the designers to a few of those behaviors may be disruptive during early design.

**Contribution 2: A set of features to support informal software design at the interactive whiteboard.**

I have demonstrated the technical feasibility of supporting the aforementioned design behaviors with the implementation of a research prototype, Calico. In order to preserve the fluid and flexible nature of the whiteboard, the first prototype began as a minimal sketching tool, with additional features carefully introduced such that the simplicity of the whiteboard experience was preserved as much as possible. Across two iterations, what was a large set of features was reduced to four that support all fourteen design behaviors without inhibiting any particular one.

These features are:

1. **Scraps and connectors.** Scraps and connectors help designers in two ways. First, they enable them to manipulate content. They enable designers to move and copy content on the canvas, which supports designers in performing the design behaviors pertaining to navigation. For example, designers may copy a user interface in a canvas, and use scraps to organize those alternatives in a canvas so that it is easy to shift their attention between them. Further, scraps function as a replacement to the lasso to manipulate content, with the advantage that it does not require a change in mode to manipulate sketches. Second, scraps and connectors help designers in represent-
ing several different types of notations. By taking any shape, being stackable, and supporting connectors, they support representations such as UML class diagrams, UI mockups, etc., and also gain the inherent benefits of scraps, such as being easily moved and copied. Further, designers may create their representations as sketches first, and convert these into scraps later, which supports the third design behavior, refining their representations.

An important aspect of scraps and connectors is that, in comparison to gesture-based interactions, the functionality of scraps is self discoverable. For example, the landing-zone presents itself to the user in a non-intrusive fashion while the user is drawing. Once the user triggers the landing-zone, the bubble menu presents all scrap operations as buttons around the selected scrap. Some gestures remain that may require training to discover, such as press-and-hold, and connecting two scraps with a stroke from scrap to scrap.

2. **The palette.** The palette provides the facility to store and reuse scraps. It supports software designers in two ways. First, the palette enables designers to more easily reuse their own impromptu notations. Such icons may include a stick figure in a use case diagram or a table that serves as a template. If the designer created their own artifacts outside of Calico, such as a set of pictures, they may also add those into the palette to use in design sessions. Second, it enables them to copy sketches across canvases. This is helpful, in particular, when generating a new alternative or level of abstraction.

3. **Intentional interfaces.** This feature organizes the design space into clusters and manages collaborative work between designers. Intentional interfaces replaces the grid feature from Calico Version One, and improves on it by providing features such as naming canvases and relating canvases to one another using tags. By tagging and linking canvases, designers can quickly find related sketches in nearby canvases, which helps the designers in moving between perspectives, alternatives, and levels of abstrac-
tion. The cluster view provides a bird’s eye perspective of all canvases, which helps in reviewing work. Further, clusters help designers retreat to previous ideas by making previous designs easier to find. Finally, designers may work together synchronously by occupying the same canvas, or asynchronously by working in different canvases. By moving to work asynchronously in a different canvas, a designer may spontaneously break away from a group discussion to develop a new alternative, and later present the new design idea by inviting others to the canvas.

4. **The fading highlighter.** Designers may temporarily annotate their sketches without making permanent marks by using the fading highlighter. It supports designers in performing mental simulations by allowing them to draw arrows and other annotations between parts of their sketches. By fading away after they are drawn, the designer avoids cluttering their sketches with strokes. Similarly, the fading highlighter supports designers in explaining their sketches, again by avoiding normal clutter. This feature is particularly useful to remote designers as it allows remote members in a meeting to draw attention to specific parts of a sketch in the absence of being able to physically point at it.

**Contribution 3: An extensive evaluation of these features both in a controlled laboratory setting and in the field with professional designers using Calico in their everyday work.**

In order to understand how Calico Version One directly impacts design, I used a controlled laboratory study to directly compare the design approach of pairs of computer science graduate students working on a hypothetical design problem using Calico against those using a regular whiteboard. Three major insights resulted from this study: (1) a subset of the design behaviors occurred in both the whiteboard and Calico condition, and, when using Calico, participants used Calico’s advanced features to perform them, (2) representative design practices happen with Calico, which was evidenced by comparisons of the structure of
the design conversations of the pairs using Calico compared to professional real-world groups in past work by Olson et. al. [57], and (3) designers that cannot write down their idea will spend energy retaining their idea and ignoring their partner until they can write it down.

Given that two hours is too short a time to become accustomed to Calico’s advanced features, and given that collaborative design behaviors may not appear when only two designers are present, I conducted a longer-term field deployment of Calico. This evaluation followed three software development teams using Calico Version Two in their own work, including a team at a commercial open source software company, a team at an interaction design company, and a team at a distributed research group. The evaluations revealed insight into how Calico was used in three very different settings, and how well it performed when encountering such diversity. It also demonstrated that Calico does indeed support all design behaviors, and that real world groups can successfully use it as part of their own design process to perform their work. The OSS group created narratives, depicted across multiple canvases, explaining how data is moved through their system. The interaction design group invented their own notations, refined their representations, and generated dozens of different perspectives that they used to create personas. The research group carried out design sessions with both local and geographically distributed members, explaining designs to each other using Calico and Skype.

Overall, Calico presented a minimal hurdle to new users in that they could ignore the advanced features and simply sketch, but were able to use the advanced features, such as the palette or lists, when they needed them by converting existing sketches.

The evaluation further showed that Calico positively impacts design sessions. Members from all groups sketched more than they would have on the whiteboard, the research group repeatedly accessed sketches that were months old and engaged in asynchronous work during their group meetings, and the OSS group, in particular, invented new ways to adapt Calico to their work. While the use of multiple canvases in intentional interfaces led to more overhead
in finding sketches, because designs were diffuse, users that invested effort in organizing what they needed could quickly locate their sketches and recalled the narratives they created across multiple canvases. Users moved between synchronous and asynchronous work when using multiple tablets and also explained their sketches to remote collaborators, using the fading highlighter to clearly underline, circle, or trace over sketches in their explanations.

The evaluation also revealed several limitations of Calico’s design. Juxtaposing sketches across canvases was difficult, users found navigating intentional interfaces sometimes not as straightforward as they felt it should be, and scraps provided generic support such as moving and copying, but stopped short of being an actual notation with full semantic support. Also, the setup was not lightweight in that it required an administrator to setup the system. Despite these limitations, overall Calico was used to perform non-trivial design tasks that used its features to perform a majority of the design behaviors in each group.

**Contribution 4: The construction of an extensible, sketch-based environment for early-phase software design.**

The tool and all of its source code is freely available to the community to build on and use in their own work. The source code is hosted on GitHub under the permissive BSD license. It can be extended by the development of plugins (indeed, some of Calico’s own features are developed as plug-ins). As more design behaviors are discovered, further features can be built into Calico to support them. Proposals of such features, such as importing software artifacts from an existing repository, are mentioned in the following section.

**Summary**

In sum, my contributions are:

1. The establishment of design behaviors as a necessary quality to support in informal design at the whiteboard.
2. A set of features to support informal software design at the interactive whiteboard.

3. An extensive evaluation of these features in a controlled laboratory setting and in the field with professional designers using Calico to perform their everyday work.

4. The construction of an extensible, sketch-based environment for early-phase software design.

9.1 Directions for future work

This section illuminates four potential areas of study to extend the work I presented in this dissertation.

9.1.1 Compositional notations

The first area of future work is to provide domain specific support for scraps. While each group in the evaluation of Calico Version Two did not strictly adhere to the grammar and syntax of formal notations, they did create sketches that borrowed idiomatic pieces from various formalisms, such as boxes-and-arrows from UML class diagrams, structured indentation in pseudo code, groupings in Euler notation, etc. In many cases, groups used scraps to represent these notations, such as when the OSS group used scraps and connectors to depict UML class diagrams and their relationships, the research group used scraps and connectors to create process flow diagrams. However, in both cases, scraps and connectors fell short in fully supporting these notations. Both groups, for instance, added cardinality to connectors using plain sketching, but the cardinality annotations were left behind when scraps were moved. There is an opportunity in this case, and others like it, to provide more support for the notational convention and semantics. In essence, when the user starts thinking of ele-
ments as a UML element, piece of code, or UI button, they desire support for manipulations that match how they think about the sketched elements.

There are several ways to provide this support. A naive approach would be to allow designers to “promote” a scrap into a specific type of scraps, such as a “UML class diagram scrap” or a “user interface scrap”. List scraps are an example of such as approach, in which an existing scrap can be changed into list scrap to vertically order a set of items. However, this approach does not fully take into account the behavior of refinement of sketches observed in design behavior 3, and may prematurely commit a designer to all aspects of a given notation, when they may only want to borrow a part of that notation.

This leads to a refined approach, which is to allow designers to add parts of notational conventions to existing scraps. I call this approach of adding parts of notational conventions to scraps “compositional notations”, in which a scrap’s behavior is determined by elementary notational behaviors that are composed into more complete notations. For instance, cardinality, coloring, and different types of arrowheads are all examples of elementary facilities that can be reused across various standard notations that we know (e.g., UML, ER, etc.). Other elements, such as a list title, bounding box, and UI widgets are similar examples of notational elements that could potentially be used as compositional elements. By allowing the user to build the level of formality of their scrap piece-by-piece, the user can use only those elements that they need, and refine the appearance and behavior of their scraps over a period of time, without enforcing all of the rules of the eventual notation too early. Clearly, there is overhead in having users manually add compositional elements for each instance in which they wish to use scraps to behave as more complete notational conventions such as UML class diagrams. However, the palette could play an important role in capturing intermediate types of scraps and allowing drag and drop refinement for type-specific behaviors.

The compositional notations feature is based on the observation that many formal notations, despite being different, share some parts of their notations. For instance, certain arrows are
universally regarded as connectors in UML, Entity-Relationship, and activity diagrams, to name a few. As another example, shape identifies the type of element in both ER Chen notation and activity diagrams. My intuition, then, is to make available these shared, elementary parts as building blocks, so that they can be composed into notations when and where the designer wants.

9.1.2 Informal analysis of scraps

A second path of future work is to explore the automated feedback that could be given through a sketched design to a designer working on some problem and solution. The obstacle, in this case, is that sketching does not lend itself to automated feedback, as a sketch is typically not interpretable by anyone other than the designer(s) themselves. In the case of Calico, however, there is the opportunity to build on the scraps feature to bring feedback that is automatically provided by the environment. The key choice, however, is which kind of feedback. In the face of sketching not to be interrupted, it is clear that formal models should be ruled out – they require far too much specification before they can provide useful feedback; they simply are best left for later phases of modeling. At the same time, one should ask whether the precision that typically comes with the formal models is necessary. More specifically, scraps could perhaps be assigned “imprecise annotations” that, nevertheless, enable useful interpretation of a sketch.

The hypothesis underneath this approach is that precise values and specifications are not necessary. Rather, I postulate that during early design it is necessary to have an approximate idea of what might happen and that it is necessary to quickly vary the underlying parameters from which this approximation is derived, such that the qualities that are sought can be explored.

Figure 9.1 presents an example of how this might work. Particularly, it shows a component,
“Order Server”, accessing data from a database, both of which are represented using scraps. Each scrap, as well as the connector between them, is decorated with annotations through which the designer can provide some rough input that is then automatically analyzed. For instance, adjusting the value of the bandwidth available between the component and the database, the designer can observe the change in the number of events that their architecture may process.

![Figure 9.1: Imprecise values can be used to provide rapid feedback.](image)

We have implemented an early prototype of this as a plugin to Calico within the domain of performance analysis. We chose to explore performance analysis because the domain is relatively simple in comparison to other formal approaches (e.g., safety analysis, syntax checking, etc.). It used activity diagrams, as opposed to other notations such as sequence diagrams, to model performance as they are similar to diagrams observed in past studies of designers [6]. In order to acquire inputs to build the activity diagram, scraps could be tagged with generic values, such as CPU process, network event, database access, each of which provided generic time-based values for analysis. The feedback was presented back to the user by changing the opacity of the scraps, where the elements in a slow performing diagram became more transparent towards the end of the path. While no formal evaluation was conducted, preliminary feedback from developers indicated that they valued the ability the give rough estimates in terms of artifacts they were familiar with (such as a network or database event) to obtain some information for how their design performs. While the result-
ing feedback does not inform the designer with precise answers of the system’s performance, it does provide the designer with more insight into their design than they would otherwise have without implementing at this stage.

9.1.3 Contextualized sketching

A third path of extending the work in this dissertation is to embed information from outside sources, such as issue trackers, repositories, etc., into Calico. This area of research is motivated by the observation that much design takes place in the context of a system already under development [15, 31], with especially the code base that is already in existence frequently taken into account. The sketches created during such maintenance design sessions often explicitly represent the existing code base in an abstract manner. Sometimes, as was the case in some of the design sessions of the OSS group, the sketch is even as simple as a few unlabeled boxes that verbally are identified. Other times, the sketch contains a significant amount of detail duplicating the code base. In such cases, valuable time can be taken up in drawing these diagrams, and there is always the risk that the diagram that is drawn does not faithfully represent the code as it currently exists.

To support these kinds of sessions, Calico could be extended to create a sketchy representation of some part of a code base that can then be interactively explored, manipulated, and augmented by designers. An initial idea is that a developer, directly in Calico, should be able to point to a configuration management repository, ask for a specific version of the code for one file, and then be presented with: (1) a scrap that represents that file, and (2) a set of outgoing edges to other scraps that represent calls to and calls by other files. This set of scraps is augmented with a set of explosion points, small markers that, when tapped, expand the information available to the developers to let them navigate the code base (not unlike Code Bubbles [10]). Tapping on one of the auxiliary classes, for instance, might make it the
center and bring up its connections. Tapping a class itself may bring up its methods, tapping a method might even bring up its code. This is of course but one way in which code could be represented. Other ideas involve reverse engineering techniques that automatically identify key methods in classes and just bring those up, using a zooming and panning technique like Code Canvas [23], using slicing to quickly bring up related code, or using higher levels of abstraction such as components or services.

### 9.1.4 Design histories

Given that all sketching activity performed in Calico is captured in a usage log, there is a potential to use this information to construct design histories from sessions performed at the electronic whiteboard. While simply constructing a replay of the sequence of strokes made by designers in a session does not capture the design rationale expressed through voice, hand gestures, and notes in surrounding artifacts like Google Docs, the sketches themselves do offer valuable insight into the process taken [52]. For example, a copy of a sketch may indicate an alternative has been explored, or a large amount of erasing may be a clue that a design idea has been rejected. Calico’s advanced features provide further contextual clues, such as moving quickly between canvases perhaps indicating that the user is reviewing their work, or intentional interfaces which allow the user to explicitly call out their intent to explore a new alternative, perspective, or level of abstraction.

There are three potential directions in which to investigate design histories: (1) reviewing one’s own work, (2) capturing the design histories of professional designers in order to serve as exemplars to learn from for novice designers, and (3) provide side-by-side comparisons of design histories, such that an instructor can compare the design histories of their students.

First, design histories can be used to recover insight into the work that led up to the design presented in the final sketches. Features such as intentional interfaces help designers by
explicitly calling out canvases as alternatives or different perspectives, but they do not show the alternatives that were present in deleted or erased canvases, or that one alternative received two hours of attention, while another received five minutes. Such insight would be useful to a designer who needs to retreat to past ideas because they have reached a dead-end in their design, and need to backtrack and reconsider previously rejected design alternatives. While the infinite space of intentional interfaces addresses some of this, a more structured history view has potential to improve recall and referencing of past work.

Second, the design histories of professional designers at work would be a helpful teaching aid to inexperienced designers. Previous research has shown that novice and professional designers differ in how they arrive at a solution. For example, novice designers are more likely to fixate on a single problem, while professionals will provisionally make a decision and move on [63]. Similarly, professionals may focus on two subjects at a time, and regularly rotate between pairs of subjects [5]. By reviewing the process taken by professional designers, novice designers would have concrete examples from which they may learn. Such teaching techniques are already used in classrooms. For example, instructors have provided students with case studies of professionals conducting design, such as videos of them at work [4]. Selected portions of these design histories could be given to the students by the instructors as examples. Previous proposals of tools have proposed similar methods of reviewing design histories, such as the Design Practice Stream (DPS) tools by Nakakoji [52], which play back a history, and CogSketch [27], which is used to understand how glyphs are used across several domains.

Third, an automatically generated design history enables instructors to gain insight into the process that students took in creating their final designs. While in class, instructors can step in and provide guidance to students who become stuck. Much work, however, happens outside of class where the instructor cannot provide active guidance. Instructors often can only see the final artifact or document that students submit. However, what if an
instructor could review the design histories of students as part of how they grade a homework? This would allow them to judge whether sufficient alternatives have been explored, whether everyone has participated, and so on.
Bibliography


Appendix A

The Prompt Provided to the Design Teams

Design Prompt: Traffic Signal Simulator

Problem Description

For the next two hours, you will be tasked with designing a traffic flow simulation program.

Your client for this project is Professor E, who teaches civil engineering at UCI. One of the courses she teaches has a section on traffic signal timing, and according to her, this is a particularly challenging subject for her students. In short, traffic signal timing involves determining the amount of time that each of an intersection’s traffic lights spend being green, yellow, and red, in order to allow cars in to flow through the intersection from each direction in a fluid manner. In the ideal case, the amount of time that people spend waiting is minimized by the chosen settings for a given intersection’s traffic lights. This can be a very subtle matter: changing the timing at a single intersection by a couple of seconds can have far-reaching effects on the traffic in the surrounding areas.

There is a great deal of theory on this subject, but Professor E. has found that her students find the topic quite abstract. She wants to provide them with some software that they can use to “play” with different traffic signal timing schemes, in different scenarios. She anticipates that this will allow her students to learn from practice, by seeing first-hand some of the patterns that govern the subject.

Requirements
The following broad requirements should be followed when designing this system:

1. Students must be able to create a visual map of an area, laying out roads in a pattern of their choosing. The resulting map need not be complex, but should allow for roads of varying length to be placed, and different arrangements of intersections to be created. Your approach should readily accommodate at least six intersections, if not more.

2. Students must be able to describe the behavior of the traffic lights at each of the intersections. It is up to you to determine what the exact interaction will be, but a variety of sequences and timing schemes should be allowed. Your approach should also be able to accommodate left-hand turns protected by left-hand green arrow lights. In addition:
   
   (a) Combinations of individual signals that would result in crashes should not be allowed.
   
   (b) Every intersection on the map must have traffic lights (there are not any stop signs, overpasses, or other variations). All intersections will be 4-way: there are no “T” intersections, nor one-way roads.

   (c) Students must be able to design each intersection with or without the option to have sensors that detect whether any cars are present in a given lane. The intersections lights behavior should be able to change based on the input from these sensors, though the exact behavior of this feature is up to you.

3. Based on the map created, and the intersection timing schemes, the students must be able to simulate traffic flows on the map. The traffic levels should be conveyed visually to the user in a real-time manner, as they emerge in the simulation. The current state of the intersections traffic lights should also be depicted visually, and updated when they change. It is up to you how to present this information to the students using your program. For example, you may choose to depict individual cars, or to use a more abstract representation.

4. Students should be able to change the traffic density that enters the map on a given road. For example, it should be possible to create a busy road, or a seldom-used one, and any variation in between. How exactly this is declared by the user and depicted by the system is up to you.

Broadly, the tool should be easy to use, and should encourage students to explore multiple alternative approaches. Students should be able to observe any problems with their maps timing scheme, alter it, and see the results of their changes on the traffic patterns.

This program is not meant to be an exact, scientific simulation, but aims to simply illustrate the basic effect that traffic signal timing has on traffic. If you wish, you may assume that you will be able to reuse an existing software package that provides relevant mathematical functionality such as statistical distributions, random number generators, and queuing theory.
You may add additional features and details to the simulation, if you think that they would support these goals.

Your design will primarily be evaluated based on its elegance and clarity – both in its overall solution and envisioned implementation structure.

**Desired Outcomes**

Your work on this design should focus on two main issues:

1. You must design the interaction that the students will have with the system. You should design the basic appearance of the program, as well as the means by which the user creates a map, sets traffic timing schemes, and views traffic simulations.

2. You must design the basic structure of the code that will be used to implement this system. You should focus on the important design decisions that form the foundation of the implementation, and work those out to the depth you believe is needed.

The result of this session should be: *the ability to present your design to a team of software developers who will be tasked with actually implementing it.* The level of competency you can expect is that of students who just completed a basic computer science or software engineering undergraduate degree. You do not need to create a complete, final diagram to be handed off to an implementation team. But you should have an understanding that is sufficient to explain how to implement the system to competent developers, without requiring them to make many high-level design decisions on their own. To simulate this hand-off, you will be asked to briefly explain the above two aspects of your design after the design session is over.

**Timeline**

- 1 hour and 50 minutes: Design session
- 10 minutes: Break / collect thoughts
- 10 minutes: Explanation of your design
- 10 minutes: Exit questionnaire