Calico: An Interactive Tool for Informal Software Design
Nicolas Mangano, Thomas D. LaToza, Marian Petre, André van der Hoek

Abstract—Whiteboards serve an important role in supporting informal software design, providing a fluid and flexible medium for collaboration. Interactive whiteboards offer the potential for enhanced support for manipulating content, managing sketches, and working across distance, but little is known about how this support affects the practice of informal software design. To understand the opportunities and challenges, we first conducted a literature review, identifying 14 behaviors that occur during informal software design. We then designed an interactive whiteboard system—Calico—to support the full spectrum of these behaviors and conducted a field deployment of Calico to three groups of software designers. Through usage logs and interviews, we examined the effects of interactivity on whiteboard use across 14 design behaviors, identifying ways in which interactive whiteboards support the practices used in physical whiteboards and where they enable designers to work more effectively.

Index Terms—design concepts, human factors in software design, interaction styles, computer-supported cooperative work

1 INTRODUCTION

Interaction designers and software developers generating and refining ideas engage in informal software design, turning to the whiteboard rather than desktop software design tools for the flexibility and fluidity it provides [7]. Unfortunately, while designers wish to manipulate content in more sophisticated ways than adding and erasing strokes [12], physical whiteboards remain a passive medium lacking active support for design. In response, nearly three decades of research [38][20] has explored the design of interactive whiteboards, investigating approaches for sketch recognition [5][24][18][8][10][8] sketch management [38][39][30][29][15][22][4][14] and distributed sketching [22][16][17][35]. Yet, interactive whiteboards are not widely used in practice [19].

We set out to understand the opportunities and challenges that interactive whiteboards afford in supporting informal software design. What behaviors are important for an interactive whiteboard to support? How can interactive whiteboards effectively support these behaviors? How does supporting these behaviors impact the practice of informal software design? What challenges remain inherent in the medium afforded by interactive whiteboards?

We first conducted a review of the software design literature, identifying 14 behaviors important to support in informal design. We then designed a single, unified sketching tool—Calico—intended to preserve the fluidity and flexibility of the whiteboard, while more effectively supporting the full range of sketching, navigation, and collaboration behaviors we identified. Finally, we conducted a field deployment of Calico to three groups of designers, recording usage logs and interviewing designers about their experiences.

Our results illustrate the breadth and diversity of informal software design at the whiteboard. Designers used Calico to create a wide range of sketches (e.g., Figure 1). The contexts in which designers worked—the nature of the design problems they faced, whether they were collocated or distributed—led to different usage of the features provided. A key benefit of Calico was the infinite number of canvases it provides, allowing designers to consider more alternatives and maintain a record of their design. Enabling designers to express relationships between canvases allowed designers to consider their design at a meta-level, providing context with which to interpret and reconstruct past designs. Our results also identified behaviors that are important to more effectively support, such as juxtaposing sketches and identifying strokes in collaborative settings.

In this paper, we first review related work in systems that support software design sketching. Next, we survey the existing literature on the actions and process of software design, reporting 14 design behaviors characterizing the activity of software design. We then report a set of design principles for supporting these behaviors and present the design of Calico, embodying these principles to support software design. To evaluate Calico and better understand how interactive whiteboards can support design, we report on an extensive field deployment of Calico, highlighting the effects of using Calico on the process of software design. This paper is an extended version of our earlier paper describing Calico [26], providing a substantially extended discussion of the design behaviors and a substantially expanded report of the results of the field deployment.

---

* N. Mangano, T. D. LaToza, and A. van der Hoek are with the Department of Informatics, Donald Bren School Information and Computer Sciences, University of California, Irvine, Irvine, CA 92697 E-mail: Innangano, Ihtatoza, andrel@ics.uci.edu.
* M. Petre is with the Faculty of Mathematics and Computing, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom. Email: m.petre@open.ac.uk.
2 RELATED WORK

Decades of research into interactive whiteboards has explored a variety of approaches, including sketch recognition, sketch management, and support for distributed sketching (see Johnson et. al. [20] for a review).

Sketch recognition systems interpret a user’s strokes, translating them into a formal object. Early systems used a predefined formal notation for interpreting sketches, such as UML diagrams [5] or user interface mockups [24], using the rules of the notation to provide feedback. Later systems explored user-expandable notations [18] and increased flexibility by delaying interpretation until desired [10], sometimes even while retaining a sketchy appearance [8]. For example, FlexiSketch supports automatically delineating sketched content into objects, performing sketch recognition to interpret objects into a formal notation, optional beautification, and a library for suggesting and inserting related objects [42]. Systems such as ActionSketch have explored approaches for enabling dynamic user interactions to be sketched [22].

Many systems have explored support for managing the many and varied sketched artifacts that are produced during meetings. Early approaches organized sketched using a filmstrip [38], hyperlinks [39], or hierarchical perspectives [30]. Later work automated particular aspects of managing sketches by automatically grouping clusters of sketches in close spatial proximity [29], shrinking sketches when moved to the periphery [15], or using metaphors such as Post-It Notes to organize and relate sketches [22]. Other systems capture and present the history of interactions with a whiteboard as a tree of storyboards [4] and allow designers to navigate a network visualization of canvases [14].

Several systems have also explored techniques for supporting synchronous and asynchronous design amongst collocated and distributed designers. Tele-board [16] is a distributed whiteboard and Post-It Note tool that allows designers to generate sticky notes from remote locations, group them, and review whiteboards in a history viewer. Designer’s Outpost [22] helps communicate distributed designers’ gestures and body language using shadows on the whiteboard. Team Storm [17] allows designers to sketch in either private spaces or public spaces, enabling designers to interact with and provide feedback on others’ sketches. Gambit [35] allows designers to use a variety of devices together, including large displays, laptops, tabletops, and phones. Other systems have begun to apply ideas from crowdsourcing to sketching, examining the use of workflows in helping groups iteratively refine sketches [45].

A few studies have investigated the impact of systems for supporting design with large format displays on practice. A field deployment of Tele-Board [16] – using traditional computers rather than an interactive whiteboard – found that moving between synchronous and asynchronous modes of work allowed designers to use the system to prepare for meetings and saved time during meetings, as designers did not need to wait to sketch their ideas. Another study [19] examined the use in practice of several large-display groupware systems for informal collaboration, communication, and awareness. The study suggests the importance of supporting critical tasks, making the system’s value evident, supporting a breadth of collaboration practices, deployment in visible ways, low barriers to use, and having a core group champion the system.

Our work builds on this prior work. Calico incorporates and extends many of these ideas to support the full range of design behaviors we identified. Our field deployment extends current knowledge about how interactive whiteboards can help support work, focusing specifically on how interactive whiteboards can support the process of informal software design.

3 DESIGN BEHAVIORS

We reviewed the software design literature and identified 14 behaviors that occur during design at the whiteboard.

3.1 How Designers Sketch

Designers draw different kinds of sketches. To explore a design problem, software designers create many different types of sketches, often within the same canvas [1][9]. They may sketch, for instance, entities and relationships, interface mockups, scenarios, or architectures [7]. Sketching different kinds of sketches on a single whiteboard is fundamental to exploring the design space, as it enables designers to explore an issue from different angles, at different levels of abstraction, or in different ways. For example, the sketches in Figure 2, 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 domain concept – a traffic intersection – to help them in working through and refining elements in the UML model on the left. In turn, their work on the UML elements lead them to revise their depiction of their domain sketch of the intersection on the right. Tools that restrict designers to a single notation with set sketch types hinder design and lead to the consideration of fewer alternatives [37].

Designers draw what they need, and no more. Few sketches are created with extensive detail; rather, designers create sketches with the detail and notation necessary to help them reason in the task at hand [40] or to reinforce what they wish to communicate [40][34]. For example, the elements in the left of Figure 2 differ in the amount of detail they contain, where some contain data attributes and others contain only a name. Low fidelity sketches with few details can be created quickly and modified easily, providing designers rapid feedback in the realization of their design [7][34]. Sketches only include as much notational convention as the designer needs in a given situation [34]. 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. Too much structure imposed by a formal notation too soon
can create unconscious barriers to change, resulting in a less exploratory and less broad search for solutions [41].

**Designers refine and evolve sketches.** As designers discuss and work through an idea, they will refine and evolve their sketches with additional details to capture their decisions [33]. Figure 3 presents an example of such a refinement, which details the evolution of a diagram from a relatively simple list to a complex box-and-arrow diagram. In general, as a result of this refinement process, sketches evolve to contain increasing visual precision, enabling the designer to revise the design to fix inconsistencies this may reveal [10].

As a designer’s understanding of the design space matures, so too does the representations that they use. As the design progresses and their decisions become firmer and additional, less-critical decisions are made, designers tend to use more elements of formal notational conventions [32]. The formal elements in the third section of Figure 3 illustrate the additional decisions that these notational conventions can capture, where some arrowheads denote cardinality and bounding boxes denote some elements’ status as an entity in the model. Refinement is not uniform across a design: portions may exist at varying levels of maturity [34].

Designers sometimes appropriate one kind of sketch into another [12]. For example, the designers initially created the data model in Figure 3 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 evolution, but in working out their design in place, they re-appropriated the list to suit their needs [27]. Designers use what is readily available over re-creating a similar diagram, especially if they do not anticipate needing the original diagram later.

**Designers use impromptu notations.** Designers do not exclusively work with the notational conventions they know (e.g., UML, ER diagrams) but also 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 [12].

### 3.2 How Designers Navigate Sketches

**Designers work with different perspectives.** Software designers create many sketches through which they shift their focus among perspectives. The designer in Figure 4, for example, is working on the user interface component for a map, and may navigate to the UML model of its data model to the far left. The designer uses each 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” [34]. 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.

**Designers work with alternatives.** In a sufficiently complex software design task, designers generate sketches of competing solutions before committing to a particular choice [44]. Rather than discussing alternatives purely verbally, sketching them allows designers to manage their focus and more effectively explore alternative solutions [28]. Once created, designers can compare alternatives and weigh their trade-offs [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 [21]. Placing diagrams side-by-side helps designers in visualizing and discussing the differences between them.

**Designers work with sketches at different levels of abstraction.** 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 4 is working with a 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 [34]. Many scenarios requiring shifts of abstraction have been documented, including the design of user interfaces [9] and web pages [13].

**Designers perform mental simulations.** Software designers use mental simulations to gain insight into the consequences of their design [43]. They may need to understand how information flows among components, or inspect their design by mimicking how an end user would interact with it. Designers “interrogate” their design by testing it against hypothetical inputs and scenarios, often annotating their sketches. For example, while discussing the logic cars use in moving through intersections, the designer in Figure 5 ran his finger along the path of the map (shown by the white arrow) to mentally simulate the path that a hypothetical car might take, verbally walking through the logic the car uses as he did so. Through such mental exercises, designers can bring to light their implicit assumptions and expose flaws in the design [34].

**Designers juxtapose sketches.** In order to compare and contrast ideas, designers often juxtapose sketches spanning perspectives, alternatives, and abstractions [34]. 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. Juxtaposed diagrams help designers in reasoning how the design might work, using the knowledge gained from one diagram to help identify the omissions, mistakes, and inconsistencies in the other [34].

**Designers review their progress.** During a design session, designers sometimes pause to take a step back, away
from the design, and consider the progress they have made and what they have yet to do [27]. For example, they may return to the 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 talk amongst themselves to assess where they are.

**Designers retreat to previous ideas.** When designers become stuck or exhaust an alternative, designers reach a stopping point in the exploration of their current set of sketches. They then may choose to return to a previous state of the design (and its sketches) to start anew [43]. 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.

### 3.3 How Designers Collaborate with Sketches

**Designers switch between synchronous and asynchronous work.** Design at the whiteboard often occurs synchronously, with designers working together on a single aspect of the design [11]. Designers sometimes break away to explore an idea independently while others continue with the main discussion [16]. This often occurs when a sudden inspiration strikes, or when a designer wishes to develop a counterexample or alternative. The designers in Figure 6, 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 diagram that explores the persistent aspects of the timing mechanism.

**Designers bring their work together.** Sometimes, as a result of asynchronous work, designers need to integrate their ideas from separate sketches into a unified design. This may involve simply combining parts of several sketches or generating an entirely new design that borrows conceptual aspects from each [12].

**Designers explain their sketches to others.** When returning from independent work or when drawing on behalf of a group, designers must synchronize their mental models of the design by explaining their work to others and vice versa [12]. Explanations are often supplemented by pointing or drawing on sketches, guiding attention to specific parts of a sketch. Designers may need to verbalize their mental simulations to explain the consequences of a particular choice, clarify the meaning of a sketch, or explain their assumptions or inspiration. While explaining, designers may add to existing sketches, using annotations to guide attention and add detail, or may simply gesture over sketches.

### 4 CALICO

To enable designers to more effectively work with sketches in informal software design, we designed Calico. One of key reasons designers use physical whiteboards is for their fluidity and flexibility. Our key goals in designing Calico were thus to maintain this fluidity and flexibility – allowing designers to focus on the content of their sketch rather than the tool used to make it – while enabling users to discover interactive features that help them to design more effectively.

Building on experiences with a previous version of Calico [27], this paper presents a new system redesigned and reimplemented from scratch to support not 4 but 14 distinct design behaviors. To make manipulating content more fluid, we introduce selection scraps and posthoc scrap creation, make scrap interactions more discoverable through bubble menus, and introduce text and list scraps. To support more effectively working with and navigating between perspectives, alternatives, and abstractions while performing mental simulations, juxtaposing, reviewing progress, and retreating to past ideas, we introduce the cluster view. To support more effective collaboration with sketches, we enable synchronous and asynchronous collaboration across multiple devices and introduce the fading highlighter to help designers explain sketches. Table 1 lists each design behavior and an associated design principle, which we then used to guide the development of Calico’s features. In the following sections, we describe Calico’s features for sketching, navigating sketches, and collaborating on sketches, and briefly report on Calico’s implementation.

### 4.1 Sketching

As in a physical whiteboard, the most prominent feature of Calico is an open canvas, allowing designers to immediately create a stroke simply by dragging their pen. Designers can select pen color, stroke width, and pen modes, and may erase strokes, undo, and redo.

A central benefit of an interactive whiteboard is the interactivity it affords – the ability to move, copy, rotate, and resize. Drawing tools often enable this through modes, allowing users to toggle between drawing and selection modes. However, modes distract from the fluidity a whiteboard provides – designers can no longer stay focused on the design task at hand and must instead maintain awareness of and actively switch among modes.

To minimize this distraction, Calico provides a lightweight selection and manipulation mechanism, allowing designers to select regions of content by circumscription, creating a selection scrap (Figure 7b). When a stroke is sufficiently long, a landing zone appears (Figure 7a); ending the stroke inside creates a selection scrap. Calico also enables scraps to be created from existing strokes, either to recover if the user has missed the landing zone or to promote previously created content into a scrap. Pressing-and-holding the pen inside a stroke that circumscribes an area triggers a dotted red circle to appear, which can be tapped to create a scrap. Scraps are inspired by Translucent Patches [23], which allows users to denote an area as a group. Scraps are movable, copy-able, deletable, rotatable, and resizeable, using the bubble menu surrounding the scrap (Figure 7b).

When a selection scrap loses focus, it immediately disappears and returns its content to the canvas, providing
Design principles for supporting the informal design behaviors of software designers

<table>
<thead>
<tr>
<th>Design behavior</th>
<th>Design principle</th>
<th>Related Calico features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sketching</td>
<td>Provide a unified abstraction for informally modeling a range of notations.</td>
<td>Scraps &amp; connectors</td>
</tr>
<tr>
<td>Designers draw different kinds of sketches.</td>
<td>Enable designers to draw exactly what they need rather than forcing content through a formalism.</td>
<td>Scraps &amp; connectors</td>
</tr>
<tr>
<td>Designers draw what they need, and no more.</td>
<td>Enable designers to incrementally and iteratively change elements over time to look and behave as different notations.</td>
<td>Scraps &amp; connectors</td>
</tr>
<tr>
<td>Designers refine and evolve sketches over time.</td>
<td>Enable designers to easily adopt notations by drawing shapes free-hand following a visual convention or by storing and reusing elements.</td>
<td>Scraps &amp; connectors, palette</td>
</tr>
<tr>
<td>Designers use impromptu notations.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Navigating sketches

| Designers work with different perspectives. | Enable designers to mix notations on a single canvas and to track relationships between multiple canvases providing different perspectives on the same topic. | Scraps & connectors, cluster view |
| Designers work with alternatives.         | Enable designers to rapidly explore alternatives by duplicating and editing elements, reusing existing elements, and duplicating and editing canvases. | Scraps & connectors, palette, cluster view |
| Designers work with sketches at different levels of abstraction. | Enable designers to manipulate and group elements into entities at various levels of abstraction and to track relationships among canvases providing views at different levels of abstraction. | Scraps & connectors, cluster view |
| Designers perform mental simulations.     | Enable designers to make transient annotations on canvases without editing the content. | Fading highlighter              |
| Designers juxtapose sketches.             | Enable designers to move elements next to each other within a canvas and to compare canvases. | Scraps, palette, cluster view   |
| Designers review their progress.          | Enable designers to step back and see the overall structure of the design by examining multiple canvases and the relationships between them. | Cluster view                    |
| Designers retreat to previous ideas.      | Enable designers to return to canvases used with previous ideas.                   | Cluster view                    |

Collaborating with sketches

| Designers switch between synchronous and asynchronous work. | Enable groups of designers to seamlessly transition between working together synchronously with a single canvas and working apart asynchronously with multiple canvases. | Cluster view |
| Designers bring their work together.          | Enable designers to view multiple canvases and move and combine elements across canvases. | Cluster view, palette |
| Designers explain their sketches to others.   | Enable groups of designers to draw attention to portions of canvases through shared, transient annotations. | Fading highlighter |

**Table 1**

interactivity benefits without forcing content to be a persistent object. To permanently retain the scrap, users may tap either of the two scrap icons in the upper left of the bubble menu to transform it into a regular scrap (indicated with a blue background – Figure 7c), either retaining the original shape or creating a neater rectangle.

Once made a regular scrap, a scrap becomes a group that is manipulatable (as described before), stackable, and connectable. For example, the ATM scrap in Figure 7d was first simply three letters drawn on the canvas, circumscribed by the stylus to create a selection scrap, and then made a regular scrap. Moving a scrap to a position where it is entirely enclosed within another scrap’s boundaries attaches it to the other scrap, allowing users to quickly create a *stack* (thereby creating hierarchically composed groups), as one would a pile of papers. Continuing the example, the Deposit, Withdrawal, and CheckBalance scraps are stacked on the Transactions scrap; moving “Transactions” moves the entire stack. Dragging a scrap off a stack ungroups it. For example, moving the scrap labeled “Deposit” from its current location to “User Interface” re-parents it to the new scrap. Scraps do not slide under other scraps; dragging a scrap implicitly moves it to the front.

Dragging the pen between scraps highlights the pen stroke, presenting the user with an option to transform the stroke into a connector, through an ignorable button. As with scraps, this can also be done retroactively by
press-and-holding a stroke that starts in one scrap and ends in another scrap. Connectors preserve the shape of the stroke, but are decorated with an arrowhead. Unlike strokes that are simply drawn, connectors are persistent and anchored to scraps: moving a scrap resizes the connector.

List scraps enable users to organize a stack into a vertical list, whose boundaries are automatically updated (Figure 8). Promoting a stack into a list organizes the immediate children of the parent scrap into a vertical list. As with the implicit grouping of regular scraps, dragging a scrap onto a list adds it, refreshing the automatic layout, and dragging it off deletes it from the list. List items also gain an associated box that can be checked and unchecked. Lists can be nested to create multi-level hierarchies.

Text scraps enable users to create typed content quickly from the keyboard, simply by pressing the enter key and typing. If a list scrap is selected, the text scrap is appended. Calico also enables scraps to be created from images.

Calico provides a palette, allowing designers to save a scrap for reuse (Figure 9). Dragging a scrap from the palette to the canvas creates a copy of the scrap. The palette is global to all canvases and users, enabling scraps to be shared.

4.2 Navigating Sketches
Calico allows designers to create and work with multiple canvases. While working in a canvas, tapping “new canvas” or “copy canvas” navigates to the new canvas and allows sketching to continue, either with a blank canvas or a canvas with duplicated content. Calico also provides a history stack with buttons to navigate forwards and backwards. Designers may choose to name their canvas with a title.

When designers create many canvases, the set of canvases may become unwieldy. To organize canvases, Calico provides a three level hierarchy: a wall, clusters, and canvases. The wall provides a zoomable, high-level grid view of clusters, allowing designers to move between separate spaces for a project or person (Figure 10). Dragging a canvas between clusters moves it, allowing users to create new clusters. Empty clusters are automatically deleted.

Tapping a cluster invokes the cluster view (Figure 11) providing a zoomable overview of a group of canvases. Clusters automatically arrange canvases into a radial layout, ordering canvases along concentric circles. In preliminary testing, users reported that clusters provided a meta-design space and wished to organize canvases as part of their design process. Calico thus allows canvases to be manually repositioned, pinning their location.

In a previous version of Calico, users navigated between canvases using a single grid. While users tended to cluster related content together, due to fixed spatial constraints, this did not scale, losing the relationships between canvases and creating a “neighbor knowledge awareness problem” [12]. To enable users to explicitly construct a narrative describing the relationships between canvases, in addition to clusters Calico supports tagging. When a new canvas is created, users are prompted to tag the canvas with its relationship to the previously visited canvas (Figure 12). The tag panel is populated with a set of tag types drawn from ways in which designers have been found to relate sketches, including alternatives, perspectives, and abstractions. The user, however, may add, edit, or delete types of tags. After choosing a tag, the new canvas is linked to the previous canvas in the cluster view, with a label denoting the tag (Figure 11). Repeatedly creating and linking canvases forms a graph structure in the radial layout.

Calico also helps users to find canvases. Navigation history is recorded, and the most recently visited canvas is highlighted with a blue halo in the cluster view (left side of Figure 11). The breadcrumb bars at the top of the canvas and cluster views (Figure 13) also let designers directly navigate to any canvas within the hierarchy.

4.4 Collaborating with Sketches
Calico supports collaborative work across multiple devices, allowing multiple designers to work synchronously on the same canvas or asynchronously on different canvases. This allows designers working in a group to branch off to their own canvas, preventing designers from “spin[ning] their wheels” while others have the floor [31]. Calico allows any user to copy or create a new canvas, work asynchronously, and later invite others to visit the new canvas. Canvases can also be shared by email or by generating a unique URL.

A fading highlighter allows users to draw temporary marks immediately visible to all users currently viewing a canvas. Marks disappear after 4 seconds. This enables designers to annotate sketches during mental simulations, reviews of progress, and explanations, particularly when working in a group with multiple devices or distributed across locations.

4.5 Implementation
Calico is implemented as a Java application, spanning approximately 100,000 lines of code. Calico’s user interface is built on the Piccolo UI toolkit for zoomable interfaces [3]. Calico uses a client-server architecture: a single server manages the persistent state of all canvases, and clients connect to the server to broadcast sketch input and receive updates from the server. In our usage, a single Calico server supported up to 20 users sketching simultaneously before performance was severely impacted. The Calico client is portable: we have used Calico with Windows desktops connected to electronic whiteboards, Windows tablets, and Windows and Mac laptops. Calico is open source¹ and freely available².

5 Field Deployment
To evaluate Calico and explore the opportunities and challenges in supporting informal software design with

¹ https://github.com/uci-sdl/Calico
² http://sdl.ics.uci.edu/research/calico/
interactive whiteboards, we conducted a field deployment of Calico.

We deployed Calico to three groups. In the research group (which included an author not associated with Calico at the time), three researchers designing a software development IDE used Calico for over a year, 6.5 months of which was included in the study period. The group was geographically distributed across two sites, but also made extensive use of Calico during a one-week collocated period. In the interaction design group (which we abbreviate as the IxD group), two designers at an interaction design firm used Calico over a five-day period. The IxD group used a version of Calico that did not contain the cluster view (including only a two level hierarchy with a grid and canvases); we thus do not report on their use of the cluster view. In the open source software group (which we abbreviate as the OSS group), five software developers at a healthcare open source software company used Calico for a four-week period.

The research, IxD, and OSS groups were setup with two Hitachi Starboard FXDUO77 whiteboards (adjacent in a room), one Hitachi Starboard FXDUO88, and one Hitachi Starboard FX, respectively. The research group had their boards installed in a wall of a shared conference room (Figure 14a) where two of the three participants were located; these participants also had access to tablets. The third, geographically distributed participant had access to Calico through a tablet and through his laptop. The IxD group had their board installed in a conference room (Figure 14b) and had access to pen-based tablets. The OSS group had their board installed in an area adjacent to the developers’ desks, containing two couches and a physical whiteboard (Figure 14c) and had access to pen-based tablets. Each group also had access to a dedicated server instance of Calico and to a traditional physical whiteboard. During the study period, we collected usage logs of Calico, recording the complete history of designers’ interactions with Calico.

To analyze this data, we first used the logs to probe designers’ use of Calico, examining instances both where usage was aligned with the design behaviors and which indicated another intention. After the study period was concluded, we conducted semi-structured interviews with designers in each group, focusing on memorable design experiences with Calico, explanations of interesting behavior observed in the usage logs, obstacles or surprises designers perceived in their use of Calico, how they felt Calico impacted their design process, and perceptions of Calico’s features.

6 Results

In the following sections, we examine designers’ overall use of Calico, the types of sketches designers created, how designers engaged in design behaviors using Calico, challenges the designers experienced using Calico, and the designers’ perceptions of Calico.

6.1 Overall use

Designers in all three groups made extensive use of Calico. The research, IxD, and OSS groups created a total of 79, 20, and 40 canvases, respectively. Table 2 lists detailed counts of the use of Calico’s features by the OSS and research groups. Due to confidentiality restrictions with the IxD group, their feature use data is not available. Overall, the OSS and research groups used Calico for 45% and 54% of the days in the 4-week and 6.5 month study periods, respectively, and performed a median of 137.5 and 160.5 user actions on each of these days.

Given the choice between Calico and their traditional physical whiteboards, the IxD group exclusively used Calico while the research and OSS groups used both, more due to ease-of-access in the moment than due to a preference of use for specific tasks. While designers made use of Calico over much of the study periods, the designers made use of Calico for the longest periods of time during bursts of activity around meetings. Designers often prepared sketches the day before, used Calico intensely during the meeting, and reviewed sketches following the meeting. While much of Calico’s value came from sketching in the moment, all groups emailed images of canvases to archive their sketches (even though Calico retains the sketches). The IxD and OSS groups did not arrange canvases into separate personal spaces through clusters; the research group, which used Calico over the longest period, did.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Total use</th>
<th>Days used</th>
<th>Median use per used day</th>
<th>Max use per used day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OSS grp</td>
<td>Research grp</td>
<td>OSS grp</td>
<td>Research grp</td>
</tr>
<tr>
<td>Strokes</td>
<td>6256</td>
<td>23915</td>
<td>45%</td>
<td>38%</td>
</tr>
<tr>
<td>Scraps</td>
<td>1636</td>
<td>14178</td>
<td>29%</td>
<td>31%</td>
</tr>
<tr>
<td>Palette</td>
<td>41</td>
<td>360</td>
<td>6%</td>
<td>10%</td>
</tr>
<tr>
<td>Fading highlighter</td>
<td>513</td>
<td>212</td>
<td>19%</td>
<td>5%</td>
</tr>
<tr>
<td>Cluster view</td>
<td>1374</td>
<td>10067</td>
<td>42%</td>
<td>53%</td>
</tr>
<tr>
<td>Overall</td>
<td>10093</td>
<td>60892</td>
<td>45%</td>
<td>54%</td>
</tr>
</tbody>
</table>

Each unit of activity corresponds to a user action (e.g., drawing a stroke, resizing a scrap, switching between canvases). The overall row includes all interactions with Calico, including activity types not listed.
The research group used Calico to record meeting notes, design a software architecture, and sketch a user interface. The designers used Calico extensively during a five-day session and sporadically throughout the rest of the deployment. They reported creating two particularly important diagrams: a process-flow chart of the entire system (Figure 15a) and a flowchart for a portion of the system (Figure 15b). Both were created during the initial five-day session and continuously updated afterwards.

The ixD group used Calico on occasion to record notes and to conduct one major design activity that spanned five days of intensive use by two designers. The designers processed a set of fifty interviews, organizing the data to build a set of user personas (Figure 16). They reported that their traditional process for such an activity would be to print the faces of each of the 50 people onto note cards, physically organize and group the cards into emergent categories, write notes on the back of the cards, and take pictures of the groupings. Using Calico, they instead imported images of the fifty individuals into the palette and dropped these onto the canvas as image scraps. They were able to create new organizations non-destructively, copying and reorganizing arrangements without losing previous groupings. If inspiration struck, they were able to return to a previous arrangement, setting aside other arrangements they had created.

The OSS group reported using Calico primarily surrounding three extensive design sessions, designing the next version of a healthcare “message processing” tool. In the first session, the developers used Calico to sketch the architecture and GUI elements of a user interface (e.g., Figure 17a – 17d). In the second session, the developers used Calico to design a set of slides to explain the software architecture of their system (e.g., Figure 17e – 17k). In this case, all members already understood the architecture, but wanted to create a representation “that was easy to understand”. In the third session, a developer doing implementation work turned to Calico to design a refactoring of source code (e.g., Figure 17l).

6.2 Types of sketches created

The designers used Calico to create canvases containing a wide variety of sketch types, including flowcharts, clusters, tables, lists, UI mockups, storyboards, source code, and box-and-arrow diagrams. Due to the importance of this design behavior to sketching, we examine the types of sketches each group created at length.

6.2.1 Research group

Flowcharts. The research group worked with process flows extensively to capture important details of their design. During an intensive five-day meeting, they modeled the core of their design onto a single canvas, depicting a complex network of individual states (Figure 15a). The discussions often centered on specific workflows and designing the behavior within those workflows. In their discussions of the diagrams, they implicitly grouped portions into regions that they understood among themselves but did not record, as they were unsure how to represent them.

Clusters. Over the course of two meetings, the research group generated a set of text scraps during a brainstorming activity and grouped them into topics. They used grouping symbols such as Euler diagrams to encode groups and drew arrows between groups to indicate several types of relationships.

User interface mockups. The research group used user interface mockups to suggest and discuss changes and additions to the user interface after portions of the system had already been implemented (e.g., lower portion of Figure 1). The researchers sometimes imported screenshots of their current interface into Calico as a visual reference, adding hand-drawn sketches of additions (e.g., Figure 15c).

Tables. Designers used tables both to help in understanding the existing design and to identify corner cases. One designer engaged in an onboarding process used a table to juxtapose simplified sketches of process flows, helping to enable a more detailed examination (e.g., top of Figure 1). The process flow contained several possible paths, and he used tables to explicitly capture several of these paths. The group also used tables to examine corner cases, helping support a mental simulation of how the algorithm might be executed.

Lists. Lists – both handwritten and as text scraps – were occasionally used to record bullet points from meetings or brainstorming sessions. For example, a group member reported writing down questions about states of a diagram that he wished to investigate later (top of Figure 1). List use was not frequent, as the designers primarily used Google Docs for note taking.

Source code. In several meetings, one designer used screenshots of pseudo code and source code to discuss his implementation. He pasted them into Calico and drew call outs to explain each method call (e.g., Figure 15d).

Dendrogram. One designer created a horizontal dendrogram to explore a set of questions generated from other sketches. Beginning with a set of four questions, he expanded these into a tree, drawing arrows to link items and adding several freehand annotations.

Timeline. The designers used a timeline to record important target dates such as conference deadlines.

6.2.2 ixD group

One and multi-dimensional plots. At the beginning of their design session, the designers created a one-dimensional plot, categorizing the people they interviewed along a single dimension (Figure 16a). 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. 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 (Figures 16b and 16c).

Table. Upon first entering a canvas (Figure 16d), the designers inserted images and drew a one-dimensional line to categorize them. They then extended segments of the line into a table, assigning images to distinct categories.

Clusters. In other cases, the designers simply arranged...
image scraps into clusters, forming groups in a bottom-up fashion. For example, in Figure 16e, the designers first wrote a topic for the canvas (“Design Behaviors”), grouped images into categories, and then labeled categories (e.g., “Enjoys to explore”). After first writing topics by hand, they later converted the topics into text scraps.

**Flowchart.** In one instance, the designers organized their interview data using a flowchart (Figure 16f). From the stories they gathered in their interviews, they drew events that occurred during a customer’s purchasing experience, and noted possible options that may occur at each point. For particularly interesting events, they added an image of the person they interviewed.

**Storyboards.** The designers used storyboards to illustrate individual steps within a user interaction. Boxes indicated each step with a picture, with descriptions of the image and call-outs providing explanation.

### 6.2.3 OSS group

**Box-and-arrow diagrams.** In two sessions, the OSS group created what they termed “box-and-arrow” diagrams, helping them to brainstorm and design an architecture. In the first session (e.g., Figure 17a), they began by brainstorming the architectural components of a user interface, enumerating the main components and creating the text scraps “Source” and “Destinations”. They then experimented with different combinations of data sources for these entities, creating additional text scraps that they connected with connectors. In a second session, the OSS group used a box and arrow diagram to explore components and how events were sent between them (e.g., Figure 17e – 17i).

**User interface mockups.** Juxtaposed with a component they were designing, the designers often created low detail mockups of its interface (Figure 17a – 17d). They used Calico to “throw ideas out on the board”, exploring alternatives in different canvases (Figure 17c, 17d).

**Lists.** The OSS group used lists to organize portions of the user interface (e.g., Figure 17b) and in describing portions of their design (e.g., Figure 17j).

**Source code.** One designer used Calico to plan a refactoring task. Depicted in Figure 17l, 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. 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 stated that he needed “a space to think freely”.

### 6.3 Support for sketching

**Designers draw what they need, and no more.** Designers varied in the level of detail they used, sometimes even when drawing similar sketches. For example, the IxD group sometimes labeled the axes of plots in detail and other times in very little detail. In some situations, designers created elaborate sketches that visually encoded a wide range of information.

Interviewed about their sketches, there was often a large disparity between designers’ mental models of what they designed and what the sketches explicitly captured. While designers from the OSS and research groups had difficulty identifying the meaning of some sketches after the fact, they recalled the overall objective, which they considered more important than the details. These sketches were used to support activity while “in the moment”. For example, the OSS group expressed most of their software architectures using only boxes and arrows (Figure 17e – 17i), only rarely labeling the connecting arrows. Most design occurred verbally, and designers only added the detail required to have something to point at during discussion.

The fading highlighter played an important role in supporting low-detail diagrams for the OSS group. They used it extensively to discuss components, draw flows of data, and discuss details of software components. By using the highlighter instead of permanently adding details, they explicitly chose to preserve sketches at a low level of detail.

**Designers refine and evolve sketches.** Designers began sketches simply, evolving them over time to more complex sketches. For example, the OSS group first created a sketch containing only handwritten names. It then evolved, as the named became text scraps and connectors were added (Figure 17a). The IxD group often began with pictures of faces, which they then categorized using visual structures. For example, in one canvas (Figure 16d), they began with a single dimensional line, added categories to the line, and transformed it into a table. While they did not set out to create a table, their design process ultimately led them to create it.

Scrap played an important role in this process, helping designers to organize and manipulate content as it evolved. The OSS and research groups reported that scraps were helpful in being able to organize and arrange content. As their design grew, the research group shifted from using plain sketches with drawn arrows to using scraps with connectors. 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 elements or reorganize elements to place similar elements in close proximity. They found text scraps easier to read than their own handwriting and more space efficient. A designers in the research group reported that scraps and connectors enabled them to more easily modify their designs, leading them to encode more of the design into more visually complex sketches and discuss their designs at a deeper level of complexity in their designs. Designers rarely made large, complex structures a single regular scrap; but scraps played an important role in creating and manipulating individual elements within larger structures.

The research group reported that, compared to a whiteboard, Calico provided “lots of space” and, for example, made it possible to draw a table that was “very clean” because they did not have to redraw things. But, as their design grew, they encountered issues with available space within a canvas. Not wishing to break up their design across multiple canvases, they reported “throw[ing] about how much detail they should put in the diagram”. 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.

Designers use impromptu notations. All groups created visual languages in their designs, devising impromptu notations to capture properties of the entities under discussion. Color was frequently used to encode an important and salient aspect of the design. To differentiate types of entities and relationships, the research group sometimes used color. For example, in Figure 15a, the black connectors represent a state transition with no information passed, blue connectors represent the passage of tokens, and red underlines denote states that were not yet implemented. Similarly, the OSS group used colors to represent particular types of data being passed. For example, in Figure 17e, 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 17f, color was used to denote how different types of data were handled. The IxD group also used color. For example, in Figure 16d, images were annotated with a colored mark indicating a category. The meaning of the notations was often not obvious and sometimes forgotten. A designer in the OSS group reported that he could not recall the meaning afterwards, but felt that it had supported his thinking during the design process itself.

Designers sometimes used the palette to record notations that could not be quickly sketched. For example, the IxD group saved and reused images of people. In order to draw a flowchart (Figure 16f), the IxD group reused elements using 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.

6.4 Support for navigating sketches

Designers work with different perspectives. All groups shifted their focus among multiple canvases representing different perspectives on their design. For example, the IxD group shifted focus between canvases categorizing their data using different visual structures (e.g., one and two dimensional plots – Figure 16a – 16c, tables – Figure 16d). All three groups found copying canvases useful, enabling, for example, the IxD group to use a template canvas to rapidly create new canvases to explore new perspectives on their data. The OSS group made frequent use of the cluster view to move between perspectives. When working with canvases, they created chains, providing an order that helped convey a story (Figure 18). This sometimes directly reflected the chronology of their exploration in the design space, while in other cases, the designers inserted canvases when they returned to previous sketches and deviated to a new idea.

Designers work with alternatives. Designers in the OSS and research groups used multiple canvases to explore multiple alternatives. In the OSS group, the alternatives were often generated as a result of conflicting opinions during discussion, inspiring a designer to copy a canvas and generate their own interpretation. For example, a member of the OSS group reported that, while working with their team members on a user interface (Figure 17c), 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 tag panel, and created a new user interface (Figure 17d). One designer reported that not being limited to a single space on a physical whiteboard meant that “more random ideas get thrown on there,” increasing the number of alternatives they sketched.

In contrast, the IxD group did not use separate canvases to explore alternatives. Unlike the other groups, the alternatives they considered were of different organizations and interpretations of data, which led them to negotiate alternatives verbally rather than through creating new canvases.

Designers work with sketches at different levels of abstraction. Designers used Calico to work with sketches at varying levels of abstraction, moving both to more and less abstract canvases. For example, the OSS group dove into the behavior of components, copying canvases, and creating new canvases at a lower abstraction level. Designers first started with a more abstract sketch of an event bus connected to event listeners (Figure 17f) before considering the design of a specific “alert” event listener (Figure 17g). They continued to produce additional sketches, exploring various components at lower levels of abstraction. For example, the “AG1” scrap in Figure 17g is represented as the “Action Group” scrap in Figure 17h, where it is elaborated in much more detail. In creating these canvases, the designers used the palette to copy elements such as “Event bus” and “Action group” between canvases. They also created canvases at a higher level of abstraction. After their exploration leading to Figure 17i, the OSS group members sketched the high level picture of messages passing through the system in Figure 17k.

Designers 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. Displaying sketches of their architecture on the whiteboard during meetings, the OSS group engaged in long discussions, gesturing at components with their hands and using the fading highlighter through a remotely connected tablet. Similarly, the research group discussed process flows, stepping through flows to simulate scenarios. The IxD group walked through user scenarios, sketching flow diagrams that described the story of the user.

The OSS group made heavy use of the fading highlighter. Displaying architectural sketches on the large electronic whiteboard during a meeting, they discussed a sketch at length, gesturing at components with their hands and using the fading highlighter from a tablet that was remotely connected to the same canvas. In one instance, they discussed a single sketch for 30 minutes, using the highlighter to depict event flows through the sys-
tem (Figure 19).

The research group made less frequent use of the fading highlighter. 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 IxD group did not use the fading highlighter. This may be because, unlike the other groups, they rarely used multiple devices, enabling the designers to physically stand next to the whiteboard. Or it may be due to differences in the types of mental simulations they performed.

**Designers juxtapose sketches.** All groups juxtaposed sketches. Designers most often preferred to create multiple canvases, comparing them by moving back and forth between canvases. In some cases, designers copied dispersed content into a single canvas. For example, a designer in the research group imported screenshots of the application’s user interface while designing his part of the system. Juxtaposed sketches sometimes served as a static reference in creating a new sketch; in other cases, designers evolved both sketches in parallel.

**Designers review their progress.** All groups reported that they reviewed their progress. Most used lists (most often handwritten or as text scraps) to summarize aspects of their design, which they sometimes referenced and updated. The OSS designers reported that they felt that lists helped them to summarize their design sessions and to reflect on designs they recently created.

Designers also reviewed their progress by rapidly moving back and forth between several canvases or by using the cluster view for an overview. While not sufficiently detailed to examine canvas content, the cluster view anchored discussion and allowed designers to gesture at canvases, with the linkages between canvases helping designers to recall “how the session played-out”. For example, the IxD 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 designers reviewed their progress, rapidly switching between the different perspectives captured in different canvases and the zoomed out cluster view.

**Designers retreat to previous ideas.** Only designers in the multi-week, long-term design sessions (the OSS and research groups) retreated to previous ideas, reporting that they did not return to previous ideas until a later design session. Both reported that, since they did not feel a need to delete unused sketches, they were able to return to old sketches more often.

In some cases, designers returned to previous sketches, continuing work. After finishing the design in Figure 17l and implementing the designed changes in code, an OSS designer returned to the large electronic whiteboard to verify that their source code correctly implemented their original pseudocode. They created a copy of the original canvas and pasted a screenshot of the newly written source code, where they continued to refine the pseudocode.

In other cases, sketches served as an archival reference. The research group originally thought that they would continuously modify their sketches, but instead used their sketches as visual references. Even this happened somewhat infrequently, as they reported that they had internalized the contents of the sketches and could reference freely without the need to have the sketches in front of them. But despite neither frequently modifying nor visiting past sketches, they reported that they still valued retaining the sketches in Calico rather than preserved through static photographs. They felt that Calico gave the sketches “a sense of permanence”, as, in the occasional cases where they did need to do so, they could return and easily make adjustments to the sketches at any time. One member reported that the “biggest benefit was crossing space and time”, as Calico provided a consistent virtual space that provided access to all canvases in a consistent location that all group members could access, and they did not need to worry about cleaning up any materials in their space or sharing whiteboard space with others meeting in the same space.

After undergoing a major refactoring, the sketches became outdated, but the research group reported that it was “good to have a reference of what the architecture was like in the previous version”. When returning, 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”.

The graph structure provided by the cluster view helped designers to locate old sessions and remember their meaning, with linked canvases assisting in reconstructing meaning. A designer in the research group reported:

“Designs get very complex... you want to keep a history of what you've done, the branches that you've pruned... 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... If you don't have the structure you'll have to create it somewhere else. [You save time] if it’s already here…”

### 6.5 Support for collaborating with sketches

**Designers switch between synchronous and asynchroneous work.** Designers in the research and OSS groups used Calico across multiple devices. For the OSS group, this led to a more informal setting in which members spontaneously broke into small groups in meetings, handing tablets back-and-forth, sketching over the diagrams, and displaying their annotations on the electronic whiteboard. With multiple tablets, multiple team members could work simultaneously without a single arbiter at the whiteboard blocking content production, an issue in whiteboard use [36]. 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 ex-
ample, while the group discussed the data that was passed from “Event Bus” in Figure 17g, one of the designers wanted to understand how the “Alert” component worked in a broader context and moved to the canvas depicted in Figure 17k to sketch a much more abstract representation, depicting how data flowed through it. After sketching it, they called the other designers over, moving back and forth between the canvases to compare them. The OSS group reported working asynchronously at least once every session and felt that it was an important benefit:

“The fact that someone can work with their own tablet or computer… is something really powerful… 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.”

The research group reported the combination of Calico and video conferencing helped them to feel more connected and enabled remote participants to be more active in sketching ideas. The remote participant reported that using a video camera pointed at the electronic whiteboards allowed him to observe the body language of the other designers. This was particularly helpful during the early phases of the collaboration, as designers 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 seeing the body language of the other designers, manipulating the same content in real-time, and seeing the others react to those edits created a sense of “connectedness.”

In contrast, the IxD group was collocated and had an established culture of working in pairs, leading them to not break into groups.

Designers bring their work together. While designers often worked separately using Calico, designers rarely combined sketches created through separate work. The IxD group did not work asynchronously, and the research group did not combine work done separately by individual designers. However, the OSS group twice combined work produced asynchronously, creating a new canvas, linking it to the previous canvases with tags, and summarizing their work.

Designers explain their sketches to others. All groups explained their sketches to one another but varied in the situations in which they did so. The IxD group worked exclusively synchronously, explaining designs only when a designer challenged decisions. The OSS group sometimes worked asynchronously and used explanations when returning to synchronous work. For example, an OSS designer took a tablet to his desk in the morning and worked out a design, creating several sketches (including those in Figure 17a – 17d). In the afternoon, he then explained his design in a group meeting to four other developers. In contrast to the other groups, the research group worked more extensively independently, leading them to more often explain work they had done separately at meetings. Designers explained their sketches by pointing, gesturing in the air, by simply talking, and, in some cases, using the fading highlighter.

6.6 Challenges using Calico

Our study revealed a number of weaknesses in Calico, ranging from usability issues to challenges inherent to interactive whiteboards. The IxD group reported that rapidly rearranging many scraps was not well supported, as the gesture of moving scraps (click and hold) felt slow to them. Due to the cluster view’s layout approach, it often zoomed out to show all canvases, making it difficult or impossible to read the content on individual canvases. This made juxtaposing sketches more challenging, forcing designers to explicitly copy canvases using the palette or to rapidly jump between canvases. It also made simply navigating between canvases using the cluster view more challenging. Designers also wanted the ability to more easily augment the set of tags, to for example, declare which alternative was chosen.

While the fading highlighter played an important role in several situations, designers often felt that they forgot to use it “in the heat of the moment”. Moreover, it was sometimes confusing which designer was drawing – designers wished to see a name associated with highlights. While the cluster view depicted the current canvas of each device, the designers still felt slowed down when moving between canvases with multiple participants, requiring that they announce what canvas they were moving to.

Nearly all groups reported that the large electronic whiteboards diminished the quality of their handwriting, forcing them to write slower or larger, write with a tablet, or enter text using a keyboard. The IxD group found the space available too small, reporting that they were “blocked by the physical limitations of the [electronic] board.”

6.7 Perceptions of Calico

The research group and OSS group both felt that, on balance, the benefits of using Calico outweighed its difficulties. The research group felt that Calico helped support their meetings. Prior to using Calico, the group used physical whiteboards and emailed pictures of the whiteboard to the remotely located team member. They preferred Calico over a formal diagramming tool as they wished to maintain informality and the ability to freely sketch. 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 physical whiteboards in their meeting spaces.

The IxD group reported that they ultimately felt that Calico did not match their needs. They wished to have infinitely sized canvases – which Calico did not provide – and felt trapped by the limited space of a canvas. Further, performance was slowed by using a large number of images on a single Canvas, making Calico less responsive.

7 Discussion

Our results reveal that, by supporting behaviors that are typically exhibited by software designers at the whiteboard, Calico can help designers to more effectively manipulate content, navigate groups & relationships
among sketches, and collaboratively design synchronously and asynchronously. 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-

<table>
<thead>
<tr>
<th>Design behavior</th>
<th>Supporting feature</th>
<th>Res. group</th>
<th>Int. group</th>
<th>OSS group</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sketching</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designers draw different kinds of sketches.</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>All groups used scraps to draw many kinds of sketches.</td>
</tr>
<tr>
<td>Designers draw what they need, and no more.</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>All groups created both low and high detail sketches using scraps.</td>
</tr>
<tr>
<td>Designers refine and evolve sketches.</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>All groups refined and evolved their sketches over time using scraps.</td>
</tr>
<tr>
<td>Designers use impromptu notations.</td>
<td>Scraps &amp; connectors, Palette</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and IxD groups created custom notations using scraps and connectors. The IxD group reused custom icons using the palette.</td>
</tr>
<tr>
<td>Navigating sketches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designers work with different perspectives.</td>
<td>Scraps &amp; connectors</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>All designers used scraps in working with sketches of different perspectives on the same canvas. The research and OSS groups tagged relationships between canvases and used the cluster view to move between them.</td>
</tr>
<tr>
<td>Designers work with alternatives.</td>
<td>Scraps &amp; connectors, Palette</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>All of the groups copied scraps to create alternatives. None of the groups used the palette to create alternatives. The research and OSS groups instead copied whole canvases, labeling them as alternatives.</td>
</tr>
<tr>
<td>Designers work with sketches at different levels of abstraction.</td>
<td>Scraps &amp; connectors, Cluster view</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The OSS group used scraps to create components at different levels of abstraction. The research and OSS groups used the cluster view to step between canvases at different levels of abstraction.</td>
</tr>
<tr>
<td>Designers perform mental simulations.</td>
<td>Fading highlighter</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and OSS groups used the fading highlighter to perform mental simulations, especially tracing the flow of data through their designs.</td>
</tr>
<tr>
<td>Designers juxtapose sketches.</td>
<td>Scraps &amp; connectors, Palette</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and OSS groups used scraps to juxtapose sketches next to each other within canvases. The research group also used the palette to copy content across canvases so that they could reference it. None of the groups used the cluster view to juxtapose canvases.</td>
</tr>
<tr>
<td>Designers review their progress.</td>
<td>Cluster view</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and OSS groups used the cluster view to review the sketches they had created.</td>
</tr>
<tr>
<td>Designers retreat to previous ideas.</td>
<td>Cluster view</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and OSS groups used the cluster view to return to past designs and design decisions.</td>
</tr>
<tr>
<td>Collaborating with sketches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designers switch between synchronous and asynchronous work.</td>
<td>Cluster view</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and OSS group occasionally used the cluster view to deviate from the group discussion and create their own canvas.</td>
</tr>
<tr>
<td>Designers bring their work together.</td>
<td>Scraps &amp; connectors, Palette</td>
<td>X</td>
<td>X</td>
<td></td>
<td>None of the groups used scraps or the cluster view to bring their work together. The OSS group once used the palette as a clipboard to combine work from multiple designers into a single canvas.</td>
</tr>
<tr>
<td>Designers explain their sketches to others.</td>
<td>Fading highlighter</td>
<td>X</td>
<td>X</td>
<td></td>
<td>The research and OSS groups used the fading highlighter to explain their designs to each other.</td>
</tr>
</tbody>
</table>
located setting. The IxD group 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. Calico supported each of the groups over many days of real work on real tasks that designers performed in service of their day-to-day responsibilities.

7.1 Use of Calico

Across all groups, at least one group performed every design behavior using Calico’s features. Table 3 summarizes the design groups’ use of features in our field deployment. While not all groups performed every design behavior using each of Calico’s features (for example only the OSS group used the palette to bring their work together), each group made major use of the features to accomplish actual work.

Based on their unique needs and tasks, each group created their own culture in their use of Calico, performing some design behaviors more than others. The OSS group created narratives, depicted across multiple canvases, explaining how data moved through their system, and, to do so, extensively used the fading highlighter to perform mental simulations and the cluster view to move between canvases depicting different levels of abstraction and perspectives. The IxD group invented notations using scraps, refined these notations, and generated dozens of different perspectives. The research group carried out their sessions distributedly, explaining designs to each other using Calico and Skype, and used scraps extensively for building process flows.

A central benefit afforded by Calico is its provision of infinite space through creating new canvases. Rather than being forced to actively manage the content on physical whiteboards, Calico enabled designers to more easily explore new perspectives and alternatives in their design. As a result, designers reported that more ideas were considered.

By giving permanence to sketches, Calico enables designers to review their progress and retreat to previous ideas more effectively through their existing sketches. Designers often took stock of their progress by using the cluster view to understand how a design session played out and examine the designs they had created. As designers presented sketches to their group, they compared the act of stepping through these canvases to giving a PowerPoint presentation. Moreover, the relationship tags between canvases also provided an important benefit in helping developers to reconstruct design history, providing important additional context in reconstructing the meaning of sketches. Even as sketches became more dated, having an archival reference of the past was important for designers. Sketching was even sometimes interspersed with implementation activities, as developers returned to design sketches after implementing them or brought in code to design implementation changes.

7.2 The Calico philosophy

Informal software design tools often support only a small subset of the design behaviors, locking users into features that exclusively support only several behaviors at the expense of support for other design behaviors. In the design of Calico, we sought to create a single, unified system for informal sketching flexible enough to support all design behaviors without major interruptions or requiring designers to switch to another tool. This enabled the users of Calico to use its features to exercise design behaviors as diverse as switching notations at will, juxtaposing high fidelity images against low quality sketches, performing mental simulations with other designers, and breaking off from groups to continue work individually.

One of the key advantages of whiteboards is their availability to immediately express an idea: designers can simply walk up to a whiteboard and sketch, without any distractions or overhead incurred by the medium. Tools such as a UML editor or wire-framing tool often sacrifice some of this immediate availability and fluidity, which may cause the user overhead in expressing their ideas in terms of the tool, rather than what is natural to them. In designing Calico, we sought to enable designers to simply sketch, adopting Calico’s additional features only when it provided an immediate benefit. Designers did not need to create sketches as scraps; they could instead transform sketches into scraps when they were ready. Designers did not need to label relationships between canvases with tags; they could simply press “copy canvas” and organize their canvases later. Regardless of how minimal the effort may be, each feature brings with it a cost that the user incurs. For example, scraps enable rapid interaction with sketched content, but require the user to think in terms of objects. Calico enabled designer to leave content as simple pen marks, never transitioning content that they did not need to manipulate as objects to objects.

Informal design is often collaborative, involving groups of designers collectively working together, who may switch between group and individual work, who may bring many devices, and who may even be distributed across locations. We sought to enable Calico to seamlessly support such group work by enabling its use across many devices, ranging from tablets to electronic whiteboards to laptops. 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. For the globally distributed research group, this enabled remote members to be active participants through manipulating shared sketches rather than simple passive contributors.

7.3 Opportunities for improvement

Usability considerations in sketching – even minor – remain an important consideration in designing systems for informal software design. Even minor performance hiccups can have a large impact on the perception of the fluidity in the whiteboard experience. A key physical limitation of interactive whiteboards is that they often reduce
handwriting quality, leading to less legible text. This suggests the need to further consider fast and lightweight mechanisms for rapid text entry such as speech to text or text recognition.

Designers constantly use general-purpose sketches to simulate and discuss scenarios, annotating and tracing paths over sketches. For example, rather than draw separate sketches for each scenario, the OSS group used the fading highlighter with the sketch in Figure 19 to trace out many different scenarios. This might be more effectively supported by allowing designers to save and reference multiple scenarios on top of general-purpose sketches through a mechanism such as persistent annotation layers on top of an underlying sketch.

One central decision in the design of Calico was the choice to break whiteboard content into separate canvases. Compared to a single, pannable infinite canvas, this has important benefits, allowing designers to easily navigate to and reference content in specific canvases. But there are also important drawbacks, as designers sometimes wish to expand a single sketch beyond the confines of its limited space. This suggests that a hybrid design may be most successful, in which content is delineated into canvases that are expandable when necessary.

As designers were able to actively work with a greater number of canvases, this played an important role in the design process. Designers often juxtaposed content from multiple canvases. While Calico supported this activity through enabling designers to copy and paste content or viewing multiple canvases in the cluster view, this support was less than ideal, as copying took time and multiple canvases were not clearly legible in the cluster view without extreme zooming. As a core activity in designing, it is crucial to support fluid and legible juxtaposition of sketches.

Due to the fluid nature of the discussions that Calico enabled, there was sometimes a need to provide greater context and awareness of these discussions. When multiple designers were sketching on the same canvas, it was sometimes unclear who was sketching what, suggesting the need for transient annotations labeling authorship. In some design activities such as reviewing progress, designers rapidly shifted between canvases. When designers were split across multiple devices, this was difficult to coordinate, suggesting a need to allow devices to temporarily subscribe to a group focus.

CONCLUSIONS

Sketching is a central part of software design, enabling designers to rapidly create and explore ideas. In this work, we made three central contributions to the state of informal software design. We identified a set of 14 design behavior patterns that together characterize the activity of informal software design and the ways in which it might be more effectively supported. We provided a set of design principles for supporting these design behaviors and implemented an interactive tool for informal software design, Calico. We then performed an extensive field deployment to three groups of designers—which spanned over 70,000 user actions—through which we uncovered ways in which Calico can help improve the practice of software design.

ACKNOWLEDGMENTS

We thank all of the designers who participated in our study. This research was funded in part by the National Science Foundation under grants CCF-1118052 and IIS-111446.

Nicolas Mangano received degrees in psychology and information and computer science from University of California, Irvine in 2007 and a Ph.D. in information and computer science from the University of California, Irvine in 2013. His research focused on understanding and supporting informal software design at the whiteboard, including in globally distributed settings and with electronic whiteboards. He won the ACM SIGSOFT Outstanding Doctoral Dissertation Award for his work on Calico. He is the co-founder of Molimir, a company whose mission is to create creative collaboration tools.

Thomas D. LaToza received degrees in psychology and computer science at the University of Illinois, Urbana-Champaign in 2004 and a Ph.D. in software engineering at Carnegie Mellon University in 2012. He is currently a postdoctoral research associate at the University of California, Irvine. His research focuses on human aspects of software development, including supporting information needs in software development, software design at the whiteboard, and crowdsourcing software engineering. He has served on several program committees and is currently an organizer of the Workshop on Crowdsourcing in Software Engineering.

Marian Petre is a Professor of Computing at The Open University in the UK. Her background includes a B.A. in Psycholinguistics from Swarthmore College and a Ph.D. in Computer Science from University College London. She received a Royal Society Wolfson Research Merit Award in recognition of her research on expert software design behaviour and reasoning. She also held an EPSRC Advanced Research Fellowship, again researching expert reasoning and representation in software design. Her research earned the Design Studies Award (2004) and the ACM SIGSOFT Distinguished Paper Award (at ICSE 2013) for ‘UML in practice’. Her research is reflected in over 100 international refereed publications, plus co-authored books including ‘Software Designers in Action’. She is a founding member of the Psychology of Programming Interest Group.

André van der Hoek received joint B.S. and M.S. degrees in business-oriented computer science from the Erasmus University Rotterdam, The Netherlands, and a Ph.D. degree in computer science from the University of Colorado at Boulder. He serves as chair of the Department of Informatics at the University of California, Irvine. He heads the Software Design and Collaboration Laboratory, which focuses on understanding and advancing the roles of design, collaboration, and education in software development. He has served on numerous international program committees, is a member of the editorial board of ACM Transactions on Software Engineering and Methodology, was program chair of the 2010 ACM SIGSOFT International Symposium on the Foundations of Software Engineering, and was program co-chair of the 2014 International Conference on Software Engineering. He was recognized in 2013 as an ACM Distinguished Scientist, and in 2009 he was a recipient of the Premier Award for Excellence in Engineering Education Courseware. He is a member of the IEEE.

REFERENCES


Fig. 1. Examples of sketches designers created using Calico.

Fig. 2: A portion of a whiteboard containing a UML diagram, a map, and annotations.
Fig. 3. A sketch that evolved from a (a) simpler notation (list) to (b) an intermediate form (boxed list) into (c) a richer notation (class diagram).

Fig. 4. Designers work with different perspectives, navigating among different types of diagrams.

Fig. 5. Designers use sketches to mentally simulate their designs in use.
Fig. 6. Designers sometimes break into asynchronous work to explore solutions, before later returning to synchronous work to share their insights.

Fig. 7. Scraps allow users to manipulate content.

(a) Landing zone  (b) Selection scrap  (c) Regular scrap  (d) Scrap stacks

Fig. 8. List scraps organize scraps into a vertical list.

(a) Palette icon  (b) Palette bar with scraps

Fig. 9. Clicking on the palette icon (a) on a scrap’s bubble menu adds it to the palette bar (b).

Fig. 10. The wall of clusters.
Fig. 11. The cluster view provides a birds eye view of canvases and enables designers to see tagged relationships between canvases.

Fig. 12. The tag panel appears in newly created canvases.

Fig. 13. The breadcrumb bar in the upper-left allows users to rapidly navigate between canvases.

Fig. 14. The interactive whiteboards and meeting space used by participants in the (a) research group, (b) interaction group, and (c) OSS group.
Fig. 15: Examples of canvases created by the research group. During their one-week intensive meeting, designers created canvases including (a) a process flow describing the main functionality of the system and (b) a process flow describing a design for testing. Designers also (c) created user interface mockups of future versions and (d) copied code into Calico to understand and revise the system.

Fig. 16: Examples of canvases produced by the interaction group to create personas. Working for interview data, the designs created various organizations of the data, including (a) a one-dimensional plot, (b) a two-dimensional plot, (c) a three-dimensional organization juxtaposed with a two-dimensional organization, (d) a table and Euler diagram, (e) spatial clusters, and (f) a flow chart. To preserve the confidentiality of the canvases, the figures were recreated, maintaining the exact visual structure but using fictional content.
Fig. 17. Examples of canvases produced by the OSS group. In the first major design session, designers focused on designing a user interface, sketching (a) a combined view of architectural components (lower left) and elements of the user interface (rest of figure), (b) a list of GUI elements, and (c) and (d) alternative GUI designs. Designers created (e) – (k) in the second major design session, focused on designing event handling for messages. A designer created (l) in order to design a code refactoring.
Fig. 18. A cluster view of canvases produced by the OSS group. The three major sessions were grouped by “chaining” canvases.

Fig. 19. An example of fading highlighter use by the OSS group (left) and a composite of 10 min (of 30) of use (right).