Calico: A Tool for Early Software Design Sketching

Nicolas Mangano, Alex Baker, Mitch Dempsey, Emily Oh Navarro, André van der Hoek
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3440 USA
{nmangano, abaker, mdempsey, emilyo, andre}@ics.uci.edu

Abstract

In this position paper, we present Calico, a sketching tool supporting early software design activities. We first provide background information about early design, including the types of models designers use and the behaviors that they typically exhibit. We then describe Calico's main features, and how they were designed to support these models and behaviors. We conclude with our experiences to date and a look at our future work.

1. Introduction

In numerous design disciplines, the role and importance of sketching are well understood and appreciated. Mechanical engineers, architects, and graphic designers, to name a few, all are known to use sketching as a way of exploring and working out initial ideas towards a proposed design [7].

Software engineers sketch, too. Indeed, they too can be found doodling on a piece of paper, or discussing a vague drawing of sorts at a whiteboard [2, 6].

One of the times when such sketching is particularly prevalent is during early software design, when a solution is not readily obvious and must in fact be "found". During this time, designers rapidly consider and evaluate many ideas, potential approaches, and alternatives, without necessarily working out each of these in a great amount of detail. It is no surprise, then, that designers of software typically abandon the traditional and more restrictive modeling languages and tools they have at their disposal, in favor of the fluid and flexible experience offered by sketching.

With a few exceptions [2, 6], the role of sketching in software design has received little attention to date. In this position paper, we briefly lay out our approach to correcting this. Specifically, we have developed Calico, a sketching tool for use on an electronic whiteboard (or tablet PC) and explicitly geared towards enabling software designers in the creative activity of early, conceptual software design. Calico provides a level of fluidity similar to the whiteboard or pen and paper, enriched with additional functionality to make the design experience more effective.

2. Approach

We draw inspiration from the general design literature, which has amassed considerable evidence on how designers behave when faced with a (complex) design problem – irrespective of which discipline they actually practice. Specifically, the following four observations drove our development of Calico:

- *Designers use low-detail in sketching designs.* The sketches are never intended to be complete, instead serving as mental aids that help designers to understand, reflect upon, and evolve the ideas that appear in their "mind's eye" [8].
- *Designers frequently shift focus.* Shifts take place among breadth of alternatives, along depth of detail [9], and between problem and solution domain [3]. The sketches produced, thus, can vary broadly and are often revisited for further revision.
- *Designers produce sketches that are ambiguous.* It is known that this ambiguity, whether consciously or subconsciously introduced, leaves room for (often key) design improvements later on [3].
- *Designers use a variety of languages in expressing their design.* Often approximations of more formal languages are used, as in box-and-arrow diagrams that resemble software architectures or data flows. Sometimes even ad-hoc and impromptu languages emerge as informal symbols that are reused across a design exercise [6].

VL/HCC Workshop: Sketch Tools for Diagramming
Herrsching am Ammersee, Germany
15 September 2008
Editors: Beryl Plimmer & Tracy Hammond

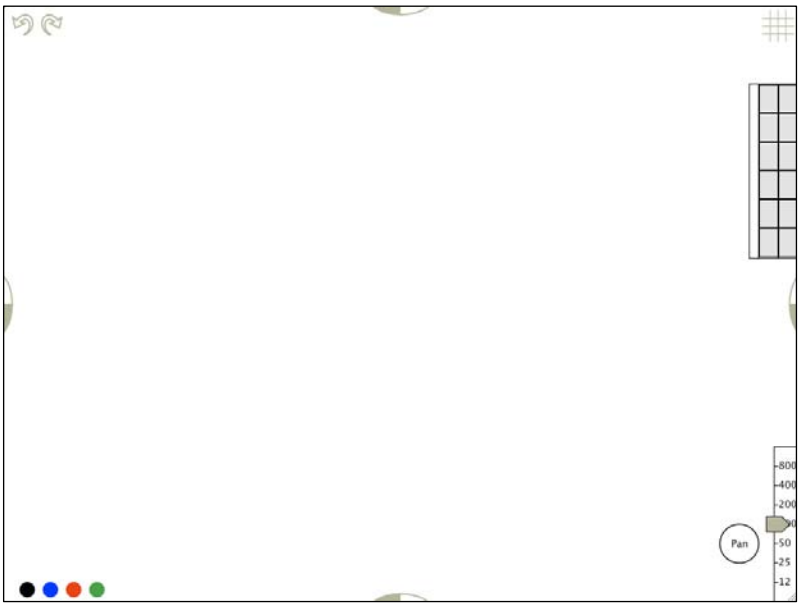

Figure 1. Calico user interface layout (palette retracts when not in use).

While these observations resulted from studies involving non-software designers, other preliminary studies indicate that software designers likely exhibit similar behaviors [2, 6]. We have therefore designed Calico specifically to amplify these behaviors and help software designers in creating, exploring, and manipulating their early software designs along these lines.

3. Calico

The basic operation of Calico is as follows. Designers work on a large canvas that is presented to them on an electronic whiteboard (or tablet PC). On this canvas (see Figure 1), they can sketch whatever they want, whether it involves figures, drawings, lists, text, annotations, or any other desired graphical or textual element. Everything they do is freehand. The basic operation of Calico therefore reduces to the same experience a designer would have on a regular whiteboard, minimally impeding the process of design they would normally follow.

Because Calico is a computer tool, however, it becomes possible to provide functional advances over a whiteboard. At a most basic level, we implemented: (1) undo and redo, (2) zooming, and (3) panning. The first feature, undo and redo, helps in ameliorating the effects of erroneous actions. The second and third provide the designer with a more malleable space in which to operate, enabling them to focus on details or create drawings larger than a single screen. These features are not the most critical, clearly, but they do provide important utility as compared to a traditional whiteboard.

The key advance underneath Calico that truly sets it apart from a traditional whiteboard is the ability to create scraps from any part of the canvas. These scraps are irregular in shape, indicated simply by a designer holding "right click" and circumscribing an area. This lifts the drawing strokes that are contained within that area from the background onto the scrap, visually assigning a certain importance to the raised region. Any raised parts of the sketch are easier to recognize and as a consequence the overall sketch can be more readily understood at a glance.

Beyond providing visual cues, scraps offer a number of other affordances. First, a scrap can be of any shape, which enables designers to assign informal meaning to differently shaped scraps. As we discussed in Section 2, informal languages can naturally emerge during design exercises. In combination with palettes (described below), scraps are the mechanism through which Calico supports the emergence of such languages.

Second, the amorphic nature of scraps, along with the free, uninterpreted strokes allowed by Calico, encourages the property of ambiguity. Generally, in early design, formal diagrams are not necessary, and in fact would hinder the design process by requiring too much detail and interrupting the creative flow [4]. Our own observations of design sessions confirm this. Numerous aspects of the drawings we have seen remain ambiguous, yet perfectly useful to the designers. Scraps support this ambiguity, since they remain informal, can be left incomplete, do not take on a "standard" shape,

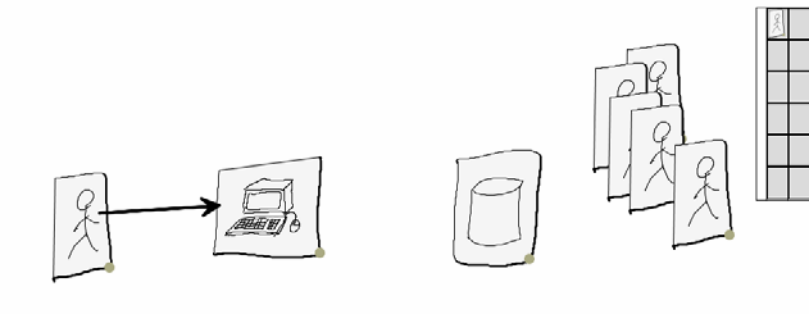

Figure 2. Scraps in Calico.

and often are used without even properly labeling them.

Third, scraps support fluid exploration of ideas and combination of ideas into design alternatives. Scraps follow the metaphor of being pieces of paper, and can be moved, grouped, related with arrows, and duplicated and erased with simple pen-based strokes that are essentially modeless (see Figures 2 and 3). Diagrams are very rapidly created and manipulated by affording the designer a level of interactivity that does not exist on the traditional whiteboard. The arrows that establish relationships are especially important in this respect. They persist with moving scraps, enabling the various kinds of box-and-arrow diagrams typically produced by software designers to be rearranged without losing their topology.

Exploration is also supported by transparent scraps. Transparent scraps enable incremental changes that do not directly modify current scrap contents. By overlaying a transparent scrap onto another scrap, a designer can make changes that can easily be rejected by removing the transparent scrap and thereby returning the diagram to its original state. If the changes are as desired, however, the contents of the transparent scrap can be dropped onto the underlying scrap to combine the two. Stacks involving multiple transparent scraps are also supported, mimicking the layering process that architects typically use in building design and we suspect to be useful in early software design as well.

Fourth, scraps form the basis for palettes. In numerous existing drawing programs, palettes containing sets of predefined shapes are provided. In Calico, palettes are initially empty (although we anticipate adding several predefined palettes) and are instead populated by a designer. The informal language of shapes that emerges during a design session can thereby be captured for use in the current session or even in future sessions. Reuse takes place simply by dragging scraps

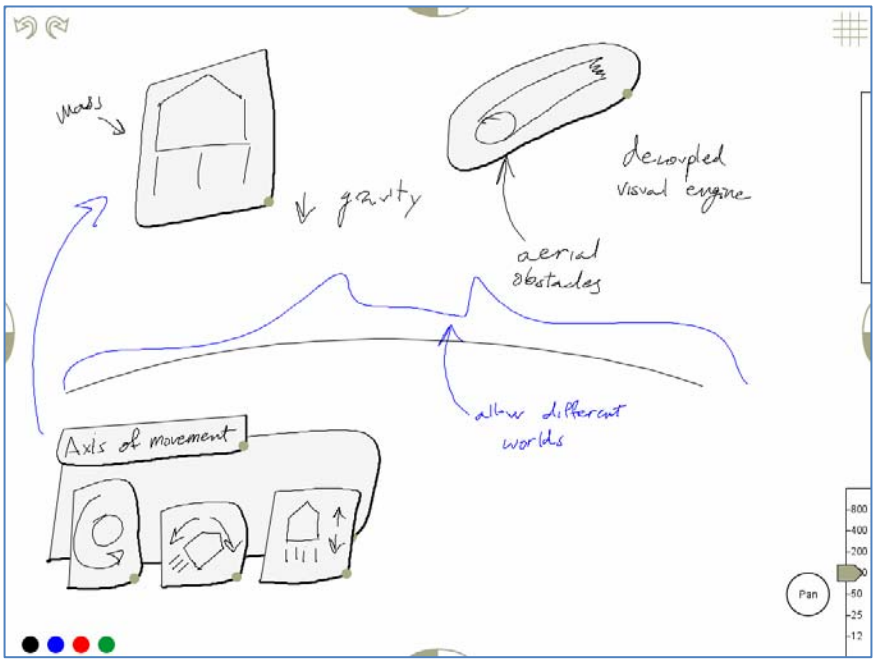

Figure 3. Calico canvas with example contents.

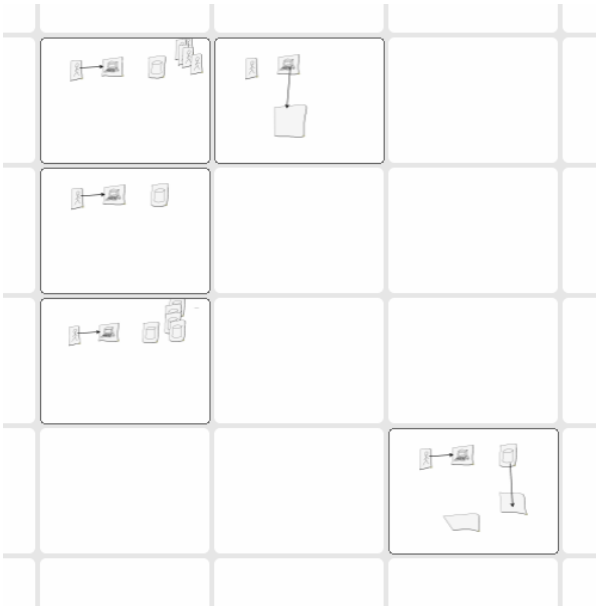

Figure 4. Calico grid (select portion).

onto the palette to store them for later use and dragging them from the palette onto the canvas to obtain a copy of the scrap (or group of scraps).

In addition to scraps, Calico provides a second advance that sets it apart. It provides the designer with a grid of canvases rather than a single canvas (see Figure 4). This grid allows them not only to explore different aspects of a design in different grid locations, but to also explore alternatives while keeping a historical trail of changes. When designing on one canvas, a simple tab allows the designer to copy the entire canvas to an adjacent grid location (left, right, top, bottom). In that grid location, they then can make any changes, while the original grid location remains unchanged. This relates to transparent scraps, but provides analogous functionality at a much larger scale. Through frequent use of canvas copying, a designer can keep a historical trail of alternatives that they have explored at a higher level of granularity, and also return to any earlier time in the exploration to start a new branch and add its explorations to the trail.

In our experience, the grid becomes a primary organizing facility for the overall design exercise. Different aspects of a design can be worked out in different regions of the grid, and each can encompass a set of mini trails representing alternatives explored for that aspect. These various aspects can then be brought together by rearranging cells in the grid, for instance through co-locating the best choices for each alternative.

4. Recording Design Histories

The fact that our approach is electronic, rather than physical, presents another potent opportunity: we can easily record detailed logs of Calico's usage. By visualizing and analyzing these usage logs we can gain insight into the process by which software design drawings were created.

This logging is implemented by recording a series of "snapshots". Whenever the user takes an action in the tool, whether it is to draw on the canvas, move around the grid, or undo an operation, Calico stores a snapshot of the resulting canvas state, with annotations about the action itself. These snapshots can then be visualized in a variety of ways. Figure 5 depicts a preliminary viewer that we are developing. It draws each snapshot as a small colored square, organized into columns representing 10-second time spans. By clicking a

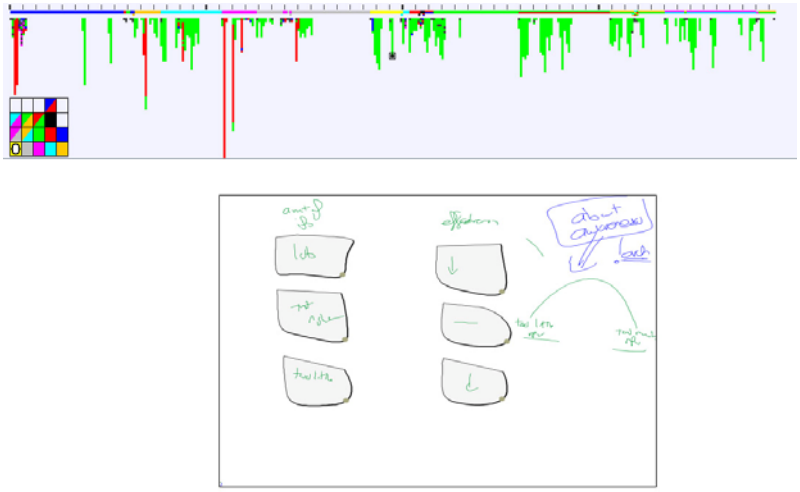

Figure 5. Calico history viewer. Timeline at top, canvas state at a particular moment in time at bottom.

square, we can see the state of the canvas, the location in the grid, and its context in the session. In addition, the design session can be moved through sequentially or animated as a real-time playback.

This ability to record and view logs provides several opportunities for future work. The log readers might be adapted to allow developers to revisit their own work or understand the work of others. The logs might be used in education; if students used Calico to design the instructor could gain insight into their process through their logs, or students could be exposed to experts' recorded design processes. Perhaps most promisingly, by deploying Calico in an industrial setting, we can gather logs from work on real-world projects and use them to understand how software designers work. We have already gathered several such logs and are currently working to analyze them.

5. Preliminary Experience

Calico has dramatically evolved since we first began building a prototype. The first two prototypes that we constructed, in fact, were unusable even by ourselves in our own design meetings. The experiences we gained from attempting to use these prototypes, however, have significantly shaped the version that we have described in this paper. This version is now in regular use in our own design meetings, and we find ourselves using most of the features (exceptions are the palette, which was only just recently included, and zooming and panning, which seem to be subject to personal preferences – in our case, we prefer to use the grid for extra space, but in small trials involving others we have seen zooming and panning used).

We now have deployed Calico to another research group at UC Irvine as well as to a local software engineering company (a second local company will follow shortly). Through questionnaires and analysis of design histories produced at the deployed sites, we anticipate obtaining comprehensive and detailed feedback on the value and use of Calico. Given that these deployments have been very recent, and that early design does not take place every day, we have to date gathered just two sessions. We anticipate being able to share some early results at the workshop in terms of analyzing these and forthcoming design sessions.

6. Related Work

Previous work in sketching can be largely classified in two categories. A first category attempts to provide a front-end to more formal drawing. Grundy and Hosking have developed a generic sketch-based front-end to the drawing of arbitrary diagrams [10], for instance, though earlier work exists by other authors that aims to translate sketched items on the screen to the formal counterparts in different kinds of domains (e.g., web pages in DENIM [14], GUIs in SILK [13], and UML in Knight [5] and SUMLOW [1]).

A second category involves tools that focus more on the creative process afforded by electronic whiteboards and sketching. Often focusing on collaboration, these tools have previously developed some features that are similar to some of those provided by Calico (as examples we mention post-it notes in PostBrainStorm [11], the ability to scrape drawings from a canvas onto a "scrap" [12], and the desire to maintain informal drawings [15].

Our work expressly does not compete with that of the first category – we are not interested in producing formal drawings from sketches. Instead, we aim to support the creative endeavor like the tools in the second category. With respect to those tools, the particular set of features offered by Calico is unique and includes what we believe are some unique capabilities (detailed scrap behavior, palettes, the grid, recording and replaying design histories) that make it particularly powerful.

7. Conclusions and Future Work

The overall goal of Calico is best described as aiming to amplify the designerly process [3]. Ideally, Calico helps software designers be more creative, consider multiple alternatives rapidly, and work in familiar ways while at the same time making it easier to work in those ways.

Calico, particularly its user interface and associated interaction model, is designed with a set of features that carefully choreograph to provide a fluid sketching experience in support of these goals. Our experiences in using Calico indicate that we now prefer Calico and the electronic whiteboard as compared to using the regular whiteboard. We look forward to the opinions and experiences from the trial deployments, and we anticipate being able to report on those at the workshop.

Our immediate future work includes leveraging the grid to support multiple designers jointly manipulating a single software design while working at multiple possibly geographically distributed electronic whiteboards. Additionally, we want to study the design histories that we will amass from our trial deployments. These provide critical opportunities to not only learn about how software designers use Calico, but also about how they approach software design problems in general. An understanding of the latter could have sig-

nificant impact on our teaching of software design and, naturally, also will feed into future Calico features.

Availability

The current version of Calico is available at the following web site: http://calico.bhnet.us.

Acknowledgments

This effort is partially funded by the National Science Foundation under grant numbers DUE-0536203 and IIS-0534775. Equipment kindly provided through a Hitachi Software Engineering America, Ltd. gift and a Hewlett-Packard Technology for Teaching Grant.

References

1. Chen, Q., J. Grundy, and J. Hosking, *An E-Whiteboard Application to Support Early Design-Stage Sketching of UML Diagrams.* Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments, 2003: p. 219-226.
2. Cherubini, M., et al., *Let's go to the whiteboard: how and why software developers use drawings.* Proceedings of the SIGCHI conference on Human factors in computing systems, 2007: p. 557-566.
3. Cross, N., *Designerly Ways of Knowing*. 2006: Springer.
4. Csikszentmihalyi, M., *Flow: The Psychology of Optimal Experience*. 1991, New York, New York: Harper Perennial.
5. Damm, C.H., K.M. Hansen, and M. Thomsen, *Tool Support for Cooperative Object-Oriented Design: Gesture Based Modelling on an Electronic Whiteboard.* Proceedings of the SIGCHI conference on Human factors in computing systems, 2000: p. 518-525.
6. Dekel, U. and J.D. Herbsleb, *Notation and representation in collaborative object-oriented design: an observational study*. SIGPLAN Not., 2007. **42**(10): p. 261-280.
7. Ellen Yi-Luen, D. and D.G. Mark, *Thinking with Diagrams in Architectural Design.* Artif. Intell. Rev., 2001. **15**(1-2): p. 135-149.
8. Ferguson, E., *Engineering and the Mind's Eye*. 1992, Ma. and London: Cambridge.
9. Goel, V., *Sketches of Thought*. 1995, Cambridge, Massachusetts: The MIT Press.
10. Grundy, J. and J. Hosking, *Supporting Generic Sketching-Based Input of Diagrams in a Domain-Specific Visual Language Meta-Tool.* Proceedings of the 29th International Conference on Software Engineering, 2007: p. 282-291.
11. Guimbretière, F., *Fluid Interaction for High Resolution Wall-Size Displays*, in *Department of Computer Science*. 2002, Stanford University. p. 157.
12. Kramer, A., *Translucent patches---dissolving windows.* Proceedings of the 7th annual ACM symposium on User interface software and technology, 1994: p. 121-130.
13. Landay, J.A. and B.A. Myers, *Sketching Interfaces: Toward More Human Interface Design*. Computer, 2001. **34**(3): p. 56-64.
14. Newman, M., et al., *DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice.* Human-Computer Interaction, 2003. **18**(3): p. 259-324.
15. Plimmer, B. and M. Apperley, *Interacting with Sketched Interface Designs: An Evaluation Study*. CHI '04 Extended Abstracts on Human Factors in Computing Systems, 2004: p. 1337-1340.