Fake People and Sticky Notes: Fostering Communication for Human-Centered Software Design

Tamara Adlin, Holly Jamesen, & Tony Krebs
Akamai Technologies, Inc.
1011 Western Ave Suite 800 Seattle, WA. 98104
+1 206 674 6000
tamara.adlin@akamai.com

ABSTRACT
Products are made or broken by the communication that exists within and between the products' creative teams. This paper describes a synergistic development process that uses a series of human-centered design artifacts to replace the single, all-inclusive specification document that so often fails to communicate product requirements to the larger development team.

The most powerful result that we've found as we've implemented our process is that communication, between both individuals and entire teams, gets smoother and expectations tend to converge, which results in a better product.

KEYWORDS
process, communication, persona, maps, interface design, specification, spec, task analysis

INTRODUCTION
In spring 1998, a small, well-staffed, talented engineering department was driving itself nuts.[i] Our process was unclear, our meetings were treacherous and exhausting, and decisions were made, then flip-flopped. Our software was good almost in spite of ourselves. The experience sparked a radical re-evaluation of our development process and organizational structure.

We built a Human-Centered Software Design team, a unified team that includes interaction designers as well as software architects and coders. We developed a simplified process model that clarified the roles of product management and development. And into that process we introduced new artifacts that better communicated the target product's design, users, and function, and promoted a common language for the product among developers, marketers, and managers.

This paper introduces you to the elements of the process we use.

THE RED BUBBLES/BLUE BUBBLES DEVELOPMENT PROCESS MODEL
Our development team required a process that balanced user needs with technology. The process we created is best expressed in the following illustration, which was originally drawn on a white board using red and blue markers [3]:

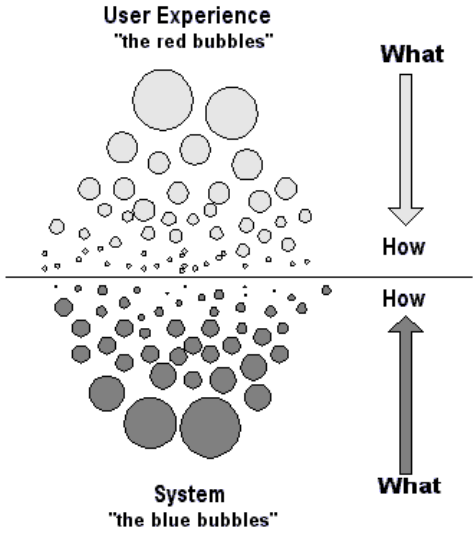

Figure 1 The Red Bubbles / Blue Bubbles process model (Color removed for this printing)

The Bubbles model requires that user experience and behind-the-scenes code development be equally valued and must go on in parallel synergy, converging on the same endpoint. It also describes our insistence on understanding overarching requirements and constraints *early* in the development process, before developers tackle decisions about how we're going to implement or code a product. Issues related to "what" tend to be large and affect the entire system (hence, the large bubbles on the edges of the diagram). Decisions on "how" to implement the design are dependant on the decisions about what needs to be implemented.

The model helps us insist on clarity from stakeholders before we start prototyping and coding, and helps the stakeholders understand the costs associated with changing requirements after the development process has begun in earnest. It also helps us identify the roles of the various

stakeholders in a project. In our organization, the Product Manager generates the Product Requirements Document (PRD), which specifies what we are going to build. The Product Manager does not, and should not, specify *how* the requirements in the PRD be fulfilled.

The premise that keeps this model intact and particularly useful is that, once the 'what we need to build' requirements are clear and communicated, we've made it difficult to change them. The process model, along with the artifacts it produces, forces decision-makers to make their decisions *early* and to stick by them.

So we have a model. So does everyone else. The Computer-Human Interaction literature is liberally sprinkled with all manner of models in shapes like waterfalls, spirals, funnels, martini glasses, and stairsteps. Models are like religions: most people have one, and they tend to be a bit ethereal and hard to understand until they are tied to actual concrete activities and artifacts.

"SPEC" IS A FOUR-LETTER WORD

Often product development starts with a specification document. It is incredibly difficult, if not impossible, to create specification one document that can successfully describe a complex system for multiple audiences. The typical spec contains far too much marketing information for the engineers and far too much engineering information for the marketers. It also contains far too little engineering information for the engineers and far too little marketing information for the marketers. No one is happy. Everybody complains and asks for more detail. The author dutifully enriches the spec, or at least enlarges it, and soon everyone stops reading it because it is huge, detailed, and hopelessly out of date.

In many organizations, the "Spec" not only has the massive primary role of describing the software that needs to be developed, but it also has the difficult (and hidden) role of creating a common and well-understood vocabulary among all of the participants. A single, massive document does not really help facilitate the communication that is so critical in a software engineering project.

Any artifact that is created to support the design, development, and evaluation of software should address the right audience with the right information at the right time. It should foster good communication and dispel confusion. It should support and be appropriate to the activities that it generates. It should include the right level of detail. The single, all-inclusive specification document (spec) needs to be replaced with a series of artifacts that are produced at various times during the development process.

THE HCD ARTIFACTS: A NEW KIND OF "SPEC"

As we drive from 'what' to 'how,' for both the red bubbles (user experience aspects of the product) and blue bubbles (system architecture and implementation), we create artifacts that digest the information we already have and help us reach our next milestone:

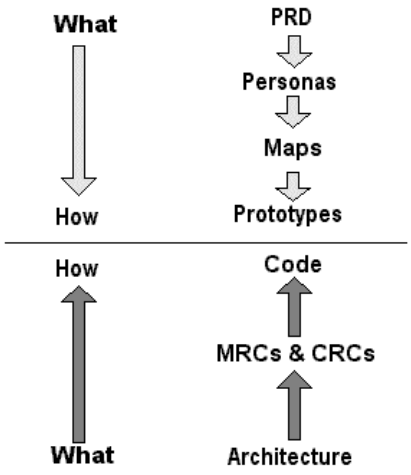

Figure 2 The Red Bubbles / Blue Bubbles artifacts

Along with the PRD, our human-centered design (or red bubbles) "spec" is comprised of the following artifacts:

- Personas
- Maps
- Information designs
- Prototypes
- Usability test plans and results

These artifacts are developed in sequence, and each depends on and evolves the information from the previous artifact. Because our 'spec' evolves with our understanding of the problem space and the user population, there is no expectation to answer all questions or completely design a software product before development starts.

There are additional artifacts—pieces of the 'spec'— that are developed in parallel by the engineers; these artifacts include architecture documents, Component-Responsibility-Collaboration (CRC) and Module-Responsibility-Collaboration (MRC) descriptions, and test code. These artifacts are primarily used to communicate among the coding engineers, and are therefore beyond the scope of this discussion.

PERSONAS

The first artifact we create is a set of personas. Because our design process is human-centered, the first thing we have to do is understand and describe the user. We present and use this information in the form of personas. For the best description of personas as a tool for human-centered software design, read chapter 9, "Designing for Pleasure," in Alan Cooper's *The Inmates are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity* [1]. Cooper's basic premise is that "Our most effective tool is profoundly simple: *Develop a precise description of our user and what he wishes to accomplish.*" (Cooper, 123) The format for this description should be a photograph and fairly detailed profile of a person who does

not exist, but who is an incredibly plausible member of the target user group.

Personas aren't real people, but they should *feel* like real people and they should *arise* from real data [1]. Good personas become 'people' you work with, the person you develop for. Much like the people in the paintings in the dorms at Hogwarts (Harry Potter's school), the personas take on a very distinctive life of their own [5].

Rather than further restate Cooper's excellent argument, it is more useful here to add what we have learned using personas as a tool in our design efforts.

- ***Create one persona per user role*:** For example, in our interactive Web broadcasting software, there are several roles: the 'staff member' is the person who is giving the presentation, while the 'audience member' is the person watching the presentation. Each has a single persona.
- ***Derive personas from information gathered about real people*:** Personas themselves are fake people, but the information used to create them is real.
- ***Never create personas by yourself*:** The human-centered design group (HCD) does the work of collecting data, writing up descriptions, finding pictures but the whole team participates in reviewing and defining the personas. The HCD group 'owns' the personas, but we rely on feedback that helps us create realistic and believable people. This increases the likelihood that personas will be adopted and will cement themselves into the entire development team's vocabulary.
- ***Publicize your personas*:** Personas only work if lots of people talk about them, use their names, and 'know' them as people. Create ways to introduce your colleagues to the personas. We have persona parties, complete with donuts, to introduce persona sets for a project. We gossip about our personas whenever possible. We look for them on the highways. Their pictures and descriptions surround us as we walk through our halls.
- ***Give your personas cars:*** This may sound silly, the descriptions must contain enough information to make the personas believable. Give them realistic hobbies and details so they are memorable.

PERSONAS HELP US COMMUNICATE

Cooper discusses the value of personas as communication and collaboration tools within an engineering team, and we've also discovered that personas are amazing tools to use as we communicate *outside* of the engineering and Product Marketing groups.

For example, for a recent project, we developed a persona named Susan. Susan was a marketing manager at a printer company, and she represented a primary user in the target market defined in the product's PRD. We showed Susan to the executive team, who had helped to describe the target market. In learning about Susan (a viable user of the product in the market they had specified), they realized that they wanted to change the target market for the product No problem. It was so early in the design process that changing the persona, and thereby the product's target user, was easy and quick. We waited for the changes to the PRD, did more research, and then developed Lewis, an Investor Relations manager. Users like Susan can still use the resulting product, but it is designed to work amazingly well for users like Lewis.

This early redirection would not have happened if we had not developed the personas with the writers of the PRD. Without the communication built on the personas, the engineering team and the executive team would have continued working with separate assumptions about the target users and the resulting product would not have worked amazingly well for anyone.

MAPS: PICTURE BOOKS THAT TELL A STORY

We have requirements in the PRD, personas representing our users, and now we need to combine the two. Our goal as a team is to find a way to view the entire user experience from end-to-end as we build our product; to tell the story of user experience simply, in a format that can help the engineers architect and build the software.

Design teams often employ task analysis in order to address this need. In general, task analysis defines user goals and breaks these down into tasks. Tasks may be further described by sub-tasks (or actions), which may also be detailed depending on the desired level of granularity. [2, 5] Hackos and Redish [4] articulate the use of task analysis methods specifically for interface design. Task analysis typically results in flowcharts that depict the task flow: first she must do this, then she must decide that, etc.

Our task-analysis takes the form of Maps. These are actual maps: they depict the entire topology, including most of the features, of an existing or planned product. They are descriptions of the user experience from start to finish, and they capture issues, questions, assumptions, and ideas for future products. They are our own version of task analysis, and they are the cornerstone of communication for our entire product development team (including product management, the HCD group, engineering, quality assurance, marketing, training, documentation, and operations).

Maps turn the PRD into a picture book. And a picture is worth a thousand (or more) words, if it is eye-catching, contains the right information for the right people at the right time. Developers don't stop to read a 200-page PRD if it's tacked to a wall—but they do stop to read our Maps.

MAPS ARE TASK ANALYSES…AND MORE

Figure 3 below provides a Map of a particular task for a fictitious product. In this map, a writer persona (Sally) prepares and delivers a document to an editor persona (Eddie), who publishes it for her. For this printing, each box is marked with an initial to indicate the color in the original map. (See below for further explanations of the color codes.)

The Maps are read from left to right and from top to bottom. The blue Post-It Notes™ at the top of each column specify subtasks. The notes that form the columns below further detail that task.

organization's goals for mapping. Blue Post-It Notes in our color legend signify known facts about a process. Green Post-It Notes are assumptions. Yellow notes represent questions and pink notes are future-oriented ideas that

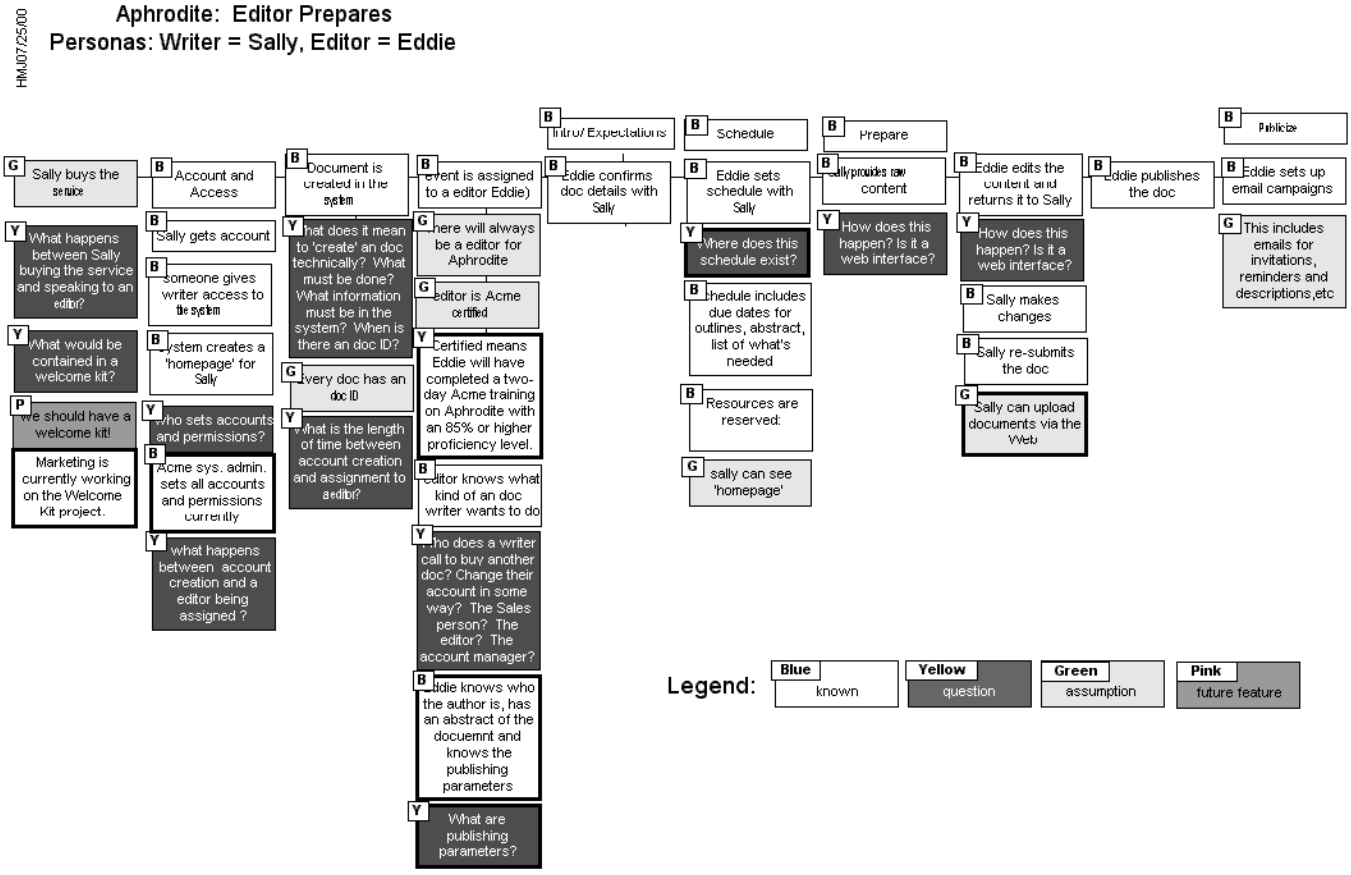

Figure 3 Map for a fictitious product named "Aphrodite"

Our Maps add several 'dimensions' to traditional task analysis:

Maps show the big picture: By the time we're done mapping, we've created a graphical representation of the entire product that can be tacked up on a wall. Anyone with a question about functionality, features, documentation, training issues or any other aspect of the product can find information on the Maps or add information in the appropriate places. Anyone working on a small piece of the product can stand in front of the appropriate Map, point, say "I am here," and understand the work that is going on around them.

Maps show what is and what could be: We create two kinds of Maps. *Reality Maps* chronicle the user experience for an existing product or process. *Design Maps*, on the other hand, are conditional. They express the way we would like a user experience to unfold. Design Maps may 'learn' from Reality Maps – retaining successful features and reworking or deleting processes that cause difficulty.

***Maps use color*:** A color legend is key to our mapping process. Color legends will be different depending upon an

should not be forgotten but are not completely relevant to the discussion at hand. Adding color-coding to the contributions on the Maps helps us to give viewers a 'quick and dirty' status report on our understanding of a process. A quick glance that leaves one with an eyeful of yellow speaks volumes about the status of the project (i.e., there are many questions that have yet to be answered).

***Maps are iterative*:** We often iterate Maps several times a day in the beginning of a project, each time inviting comments, red pens, additional Post-It Notes, and discussions as we post the Maps on the walls.

***Map creation is a group activity*:** Maps must be created by groups of people. In our organization, early versions of Maps tend to be created by members of the HCD group working with the Product Managers who write the PRD (in the case of design Maps) or actual users (in the case of reality Maps). As we discover questions and issues, we bring in various engineers to help. The engineers who join us can read the Maps, understand the context of the discussion, and add their input.

Maps demand publicity: While some task analysis charts are created for use by designers only, our Maps are created to be publicized. We post our Maps early and often, arrange meetings for the engineering team (replete with donuts and other sugar bombs) to walk through the Maps, distribute Maps to the training organization, etc. Everyone is encouraged to add to them and to use them as a guideline when building pieces of the product.

MAPS HELP US COMMUNICATE

We use Maps for all of our projects, and they have revolutionized the way we interact as a team. QA and Usability use the Maps to build their test plans. Engineers use the Maps to understand the product requirements and task flow that the code must support. Training and documentation staff use the Maps to build support materials before the coding even begins. Interaction designers use the Maps to create UI prototypes that can be tested and improved early on. Maps guarantee that all of these efforts are grounded in communally-held assumptions. Here are some of the ways Maps encourage communication.

Mapping sessions can end 'rototilling:' Rototilling is our affectionate term for the practice of discussing minute details or challenges in a system ad nauseum and thereby derailing the main discussion. A strong facilitator can put an end to such conversations during a mapping session with a robust, "HALT! Put it on a Post-It Note, stick it on the Map, and let's move on." Mapping participants quickly accept this practice, because they know that issues they raise are not lost, but are captured and will be addressed at the right time.

Maps are accessible and easy to read: Maps are always posted in common areas, and everyone is encouraged to add Post-It Notes and write in comments. These contributions are always included in the next iteration of the Maps. Because Maps are picture books for adults, it's easy to take a quick look and take away important conclusions. Big, unruly Maps with many branches often mean unnecessarily complex processes. A Map that changes over time from lots of yellow to a Map with lots of blue indicates progress in understanding the user's process.

Maps are easy to change and maintain: Whether on paper or in electronic format, it is easy to add individual comments and edits to the Maps in a timely manner.

GUIDELINES FOR BUILDING MAPS

***Create Maps on real paper*:** Although we eventually transfer all of our paper, Post-It, and pen Maps to electronic format, the tactile nature of mapping is an asset to the initial process. A large common workspace proves invaluable for this type of group activity, so we always start with large rolls of butcher paper and stacks of Post-It Notes. Participants have fun using familiar low-tech tools in a creative way. And it's easier to see the gestalt expressed in a Map that grows on a large office wall than one that takes shape on a cramped computer screen.

***Give color consistent meaning*:** Your legend should fit the context of your work and the goal of your project.

Pick an assertive but objective leader for map-making: Each participant has their own concerns that they want to describe, often in great depth. Leaders should enforce the expectation that participants *write down* their ideas or concerns and post them on the Map. The group leader also should not have a personal agenda for the results, other than getting the process mapped.

Focus on the experience, not on the technology: Try to tell a story about the experience of the persona as he or she uses the product you are designing. The Maps are not meant to be a flowchart of features; they are much more helpful when they describe a start-to-finish experience of a person with the product.

Give every participant in a mapping session the tools that they need to make contributions: All participants should have plenty of Post-It Notes in all of the colors defined in your legend and a good pen. If the group is large or it's difficult for all participants to be near the Map, it may be helpful to designate a "Post-It Note middle man"—someone to take Post-It Notes from participants as they write them and affix them to the Map.

Make one Map per major task: We usually start out with a lot of Maps, which expand as questions are answered and people write up their input. We find that as time passes, the Maps tend to get shorter and Maps tend to get combined, so the end result may be fewer Maps with clearer task progressions.

Use Visio® to maintain the maps: Once they are on paper, Maps should be put into electronic form so that it's easy to keep track of changes. We've found Visio to be a powerful tool that works well for this task.

Encourage comment: The Visio versions of our Maps hang on the walls in high traffic hallways of our office, accompanied by pens and Post-It Notes. Developers, project managers and others are encouraged to add their two cents, to write in answers to yellow Post-It Notes questions, add overlooked issues (yellow), suggest features (pink), etc. Larger organizations could use Web sites or automatic e-mail notification of significant Map changes to keep the Maps current and accessible to all team members.

Be diligent in maintaining the Maps: If participants take the time to edit the Maps, it is crucial to recognize and include these comments in a timely manner. We add red borders to all changes in the newest iterations of our Maps. In this manner, quick glances at Maps offer an easy gauge of Map status. Numerous red borders indicate lots of contributions between iterations.

CONCLUSION

Excellent software is designed as a cohesive system, not a collection of parts glued together. In order to design a cohesive system, communication between the various participants has to be stellar. In short, a group that is trying to *create* a well-oiled machine has to *work* like a well-oiled machine.

The process we use is based on a model of development that is itself tied to the organization of our company. For example, we depend on having a clear product requirements document (PRD) produced by product management. The most powerful result that we've found as we've implemented all of the tools described below is that communication, between both individuals and entire teams, gets smoother and expectations tend to converge, keeping the end user at the center of the development process.

ACKNOWLEDGMENTS

The process and ideas expressed in this paper are the work of an amazing team with a wonderful leader: Dan Gallivan. The process described in this paper is the brainchild of the authors and Dan Gallivan, Eric Robinson, Lynne Evans, Shellie Knawa, and the entire Akamai / Seattle engineering team.

REFERENCES

1. Cooper, A. *The Inmates are Running the Asylum: Why High-Tech Products Drive us Crazy and How to Restore the Sanity*. SAMS, Macmillan Computer Publishing, Indianapolis IN, 1999.
2. Dumas, J., and Redish J.C. *A Practical Guide to Usability Testing*. Alex Publishing Corporation, Norwood NJ, 1994.
3. [Gallivan, Daniel. Personal communication.]
4. Hackos, J.T. and Redish, J.C. *User and Task Analysis for Interface Design*. Wiley Computer Publishing, New York, NY, 1998.
5. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. *Human-Computer* Interaction. Addison-Wesley, Harlow, England, 1994.
6. Rowling, J.K. *Harry Potter and the Sorcerer's Stone (Book 1).* Arthur A. Levine, New York NY, 1999.

---

[i] This paper describes a process evolved by a team of developers at Netpodium Corporation, a company later acquired by Akamai Technologies.