What Designers Want:
Needs of Interactive Application Designers

Valentina Grigoreanu[1], Roland Fernandez[2], Kori Inkpen[2], George Robertson[2]
Oregon State University[1], Microsoft Research[2]
grigorev@eecs.oregonstate.edu; {rfernand, kori, ggr}@microsoft.com

Abstract

Designers' extensive software needs have not been adequately documented in the research literature, and are poorly supported by software. Without appropriate tools to support their needs, designers have difficulty knowing the best way to evolve the look and feel of interactive applications they are designing.

In order to inform the design of new tools for interactive application design, we used a grounded theory approach to find out what designers' needs are when designing such applications. This paper reports our findings (20 designer needs) from content analysis of five types of artifacts: surveys, Blend discussion list emails, Dreamweaver forum entries, Flash forum entries, and interviews with ten designers. These 20 needs were then validated in follow-up interviews and focus group sessions. The results of this work revealed trends regarding the importance of each need and show that flow is one of the most important needs.

1. Introduction

With the advent of tools such as Adobe Dreamweaver and Microsoft Expression Blend, designers are acquiring increased control in creating both the look and feel of interactive desktop and web applications. While much work exists on designers' processes and tools, we know little about the extent of the problems designers encounter with this software, and how design tools can support those needs. In this paper, we describe two grounded theory studies conducted to explore interactive application designers' needs and to validate our findings.

Jeanette Wing states "Computational methods and models give us the courage to solve problems and design systems that no one of us would be able of tackling alone" [16]. The power of computers allows designers to offload some of the computational complexity of the design process onto the computer. We believe that there is much room for improvement in software support of designer's tasks. These tasks might involve acquiring and reusing earlier work for a new purpose, or optimizing users' interactions.

Consistent with the research method of grounded theory [7], we conducted content analysis of data from several sources to identify these needs: (1) a small open-ended qualitative survey, (2) emails to a Blend discussion list, (3) posts to a Dreamweaver forum, (4) posts to a Flash forum, and (5) 1.5-hour interviews with ten designers. This study helped us better understand designers' needs.

Our resulting theory is a set of 20 needs experienced by designers while creating, iterating, and communicating their designs. We validated our findings through an interview with two experienced design managers and two focus group sessions with nine designers. The results demonstrate that designers have many unsupported needs. Each need is a starting point for the creation of better design tools. We also report which needs are likely to appear together, as well as the relative importance of each need.

2. Related Work

A *designer* is a professional who creates plans to be used in making something [6] (in our case, interactive software applications). Research on the design process suggests that many designers start out with sketches and continue with intermediate representations (e.g., maps, storyboards, schematics, prototypes, mock-ups, and specifications) [11]. We also know that designers find behavior more complicated to design than visual aspects of applications, and that communication plays a vital role in the design process [10]. However, we do not have an overarching view of the kinds of needs these designers have, which software should address.

Since the two most popular interactive application design tools are Dreamweaver and Flash [10], it is not surprising that much of our knowledge of interaction design stems from the web world. *Interactive websites* include online forms, surveys, and databases, and their design often requires end-user programming when they are not created by professional programmers (e.g., [13] and [10]). Early work in this area found that end-user

web developers understood the roles of visible components (such as fields and buttons), and used technical terms (such as "file", "database", "page link", and "member") [13]. However, they were vague in the implementation of these terms, did not understand how hidden operations worked, and did not mention constructs such as "variables" or "loops."

Designers use a variety of tools for different phases of the design, including (from most to least popular): Adobe Photoshop, Adobe Dreamweaver, Microsoft PowerPoint, Adobe Illustrator, Adobe Flash, Microsoft Visio, Adobe InDesign, Omni Group's Omnigraffle, Microsoft Visual Studio, Adobe Fireworks, Adobe Director, Microsoft FrontPage, Adobe After Effects, Axure RP, Adobe Flex, Adobe GoLive, and Microsoft Expression Blend [10]. Research platforms for web designers include: CLICK [13], WebFormulate [1], FAR [3], DENIM [12], and WebSheets [-1125383474]. Even with this multitude of tools, a designer trying to create an interactive web or desktop application still faces many challenges. We believe a possible reason for this is that little is known about the overall extent of designer needs and their relative importance, to be explicitly addressed in design tools.

The design task has been defined as a "knowledge-based problem-solving activity" by some researchers, specified by a set of functions (constraints stated by the intended users of the design and the domain itself) and a technology (the components available for design creation and the relationships between them) [4]. Chandrasekaran [4] reviews existing literature on designers to get a better understanding of the steps involved in the problem-solving task of design, and proposes the following task structure: design, verify, critique, and modify. Smith and Browne [15] break up the design problem into five elements: goals, constraints, alternatives, representations, and solutions.

Unlike these two works, we take a grounded theory approach to determining designers' needs. Relying on literature alone in determining designers' needs might lead to overlooking facets particular to designers' tasks. This paper therefore generates theory from the ground up, augmenting knowledge from related literature by collecting and analyzing designers' anecdotal evidence. In doing so, we focus on informing the design of tools for designers.

3. Grounded Theory Study Setup

3.1. Methods

We used the objectivist grounded theory methodology [7] to identify designers' needs, since it allowed us to reduce the bias of our previous experience. Our methodology was also highly influenced by Charmaz' [5] constructivist grounded theory approach. Unlike objectivists, constructivist grounded theorists pay special attention to the context of the research (the participants' and researchers' assumptions, implicit meanings, and tacit rules).

We triangulated our data and analyses on several fronts. First, since particular data sources could be biased toward revealing particular needs, we collected data (282 artifacts) from five different sources which are described in further detail in the next section. Second, since one researcher might notice different needs from another, we made sure that two interviewers were present at three of the twelve interviews. Finally, since bias may affect the results derived from the collected data, two researchers coded 40% of the data together (reaching an inter-coder reliability of 81%), before individually coding the rest.

While collecting data, we also iteratively analyzed them. Unlike traditional methods for generating theory, grounded theory calls for emerging theory to be integrated with further data collection and data analysis [8]. It also calls for the researcher to remove preconceived ideas in the early data collection phases. We therefore started with very open-ended survey and interview questions, generally asking about designers' troubleshooting, problem solving, and how they find and fix errors in designs. Our questions became more focused with every iteration thereafter, culminating in one last interview with two experienced designers which was structured around our final 20 needs, to verify and learn more about our emerging results.

3.2. Procedures and Participants

In grounded theory, data collection and data analysis are a part of the same iterative process. We describe all of the data collection methods here, and present our results in the next section.

Survey. We began with a small open-ended survey to determine designers' needs. The survey asked four open-ended questions: what tools they used, the type of design they did (visual, interaction, etc.), problem-solving or troubleshooting needs they had, and their prior experience with creating functional prototypes.

Six designers (three females) on a user experience team at Microsoft completed the survey. All six of the participants designed both the look and feel of the applications they worked on. Also, all but one had end-user programming experience: developing at least a website using languages such as JavaScript and Flash. The software used by this small group of designers varied widely, but was consistent with findings from larger surveys of designer populations [10].

Blend Distribution List. We next harvested questions from a mailing list where designers discussed design issues. Starting with 7524 emails, we narrowed the scope down to 236 by counting only initial inquiries, not the replies, and by only including emails that contained the word "designer." Finally, from that set, we analyzed the 25 emails sent within the past year, related to problems designers were having.

Dreamweaver and Flash Forums. We also harvested questions asked by designers on the Adobe public web forums for Dreamweaver and Flash, the two most frequently used interactive design application tools [10]. All forum entries were original design inquiries. We took the 25 most recent such entries from each forum, to match the number of artifacts we had collected from the Blend distribution list.

Formative Interviews. Informed by what we had learned thus far, we conducted ten qualitative interviews with designers across several product teams (four females). Interviews lasted 1.5 hours each, and participants received a lunch voucher for participating.

To encourage participants to show examples of designs they had created and problems they had encountered, the interviews were conducted at the designer's desk when possible (for eight of the ten interviews). The two others were conducted in a cafeteria and the interviewees sketched in a notebook to clarify the anecdotes they were giving.

The qualitative interviews provided us with an open-ended, yet in-depth, exploration of a topic of interest to the interviewees. As Charmaz [5] points out, many grounded theory studies use a one-shot interviewing approach, when sequential interviews provide interviewers with the ability to follow up on earlier leads. To collect a richer understanding of our designers' needs, we invited the survey respondents to also be interviewed (we interviewed all three males and one of the females). We also interviewed six designers who were new to our research to get a fresh perspective on our emerging results.

4. Grounded Theory Results

4.1. Do Designers Need Tool Support?

Responses from the survey showed that our designers did indeed need interactive application design tools to better support their problem-solving activities. This lack of support sometimes had a negative impact on the designs they created. (For all the quotes in this paper: F=female, M=male, U=unknown, S=survey, O=forums, and I=Interviews.)

> FS: *Not having tools for the kind of visual debugging help I was looking for changed my design in a negative way; finding tweaks, workarounds, giving up…*

While some environments offer some support for one type of design problem solving (end-user debugging and troubleshooting), those tools are often only about the underlying code, and are based on debugging tools used by professional developers:

> MS: *When I am in Blend/Visual Studio, I use the code debugging tools, though there is no such thing as something that helps me find out why something is showing up two pixels off.*

Designers need problem-solving tools to support their design activities at a higher level than the code (i.e., at the same level of abstraction as their design representations), and even the code features needed to be geared toward designers themselves.

Designers encounter many problems which software environments could support. We identified 20 non-orthogonal needs through content analysis of the artifacts (see Table 1), as well as the relationships between them (see Figure 1). Each need is a starting point, rather than an endpoint. The edges between two codes give a sense of the frequency with which the needs appeared together (the more often they appear together, the shorter and thicker the edge between them is). Generally, the problems fell into three categories: problems in *creating* the first version of the design, in *iterating* on it, and in *communicating* it.

4.2. Designers' Creation Needs

The codes which strongly related to "creating the first design" (and therefore perhaps also to creativity) were *propose* (and its neighbors: *look*, *feel*, and *training*) and *reuse* (and its neighbors: *extend*, *themes*, and *flow*). Some of these needs also came up during the iteration and communication steps.

Propose. Designers need tools to support *proposing a design*, since they sometimes need help coming up with that first design idea; "the right representation."

> FI: *How do you show that some items in the list are different from the other ones?*

Look. This was a very general category encompassing all needs that were about the *look* of the designed application. Designers need to be able to problem solve the look of an application, when the desired visual aspects are not achieved or could be better. The look that a designer might have in mind is not always easy to create an artifact around.

> MI: *Developers often don't get the colors right, so it takes the detailed eye of a designer to make sure the design's right (generating bits of code helps with this).*

Table 1. Designers' needs and their definitions (entries are alphabetized).

Automation	Automating redundant steps of design by applying one action to many parts of the design.
Bugs	Creating bug-free code by getting help in fixing bugs, or by generating correct code.
Cleanup	Having to clean up code or take out unused generated code.
Communication	Communicating the design to developers and others through sketches, speech, prototypes, etc.
Compatibility	Ensuring that the same application works in multiple environments.
External Constraints	Keeping track of design constraints, and helping ensure that the constraints are met.
Extensibility	Extending or customizing the functionality of the design environment.
Feel	Ensuring that the feel of the design is correct.
Flow	Representing how data, events, and other resources flow through the design.
Granularity	Switching between levels of abstraction of the application's design.
Jargon	Understanding jargon.
Look	Ensuring that the look of the design is right.
Optimization	Optimizing either the look or feel of the application.
Propose	Proposing the first version of a design.
Reuse	Reusing someone else's or one's own code and designs.
Settings	Identifying incorrect or inexistent software, hardware, or other settings external to the design.
Testing	Evaluating the design's correctness for different situations.
Themes	Creating a design theme or to applying a theme from somewhere else.
Training	Getting training resources or other help.
Usability	Evaluating usability issues in a design.

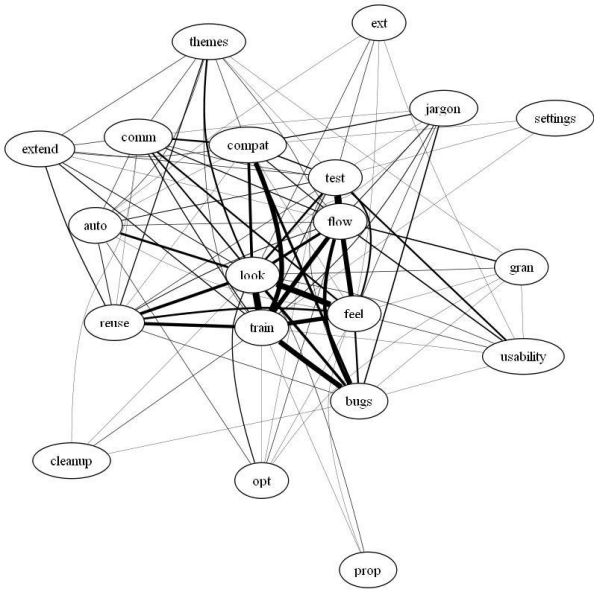

Figure 1. A constellation graph showing the relationships between the codes: the thicker and shorter an edge, the more artifacts those two codes (end-points of the edge) appeared together for.

Feel. Proposing the initial *feel* of an application is also hard. Like *look*, this was a very general need, referring to a wide range of problems having to do with a design flaw in the behavior. Examples included navigations and other interactions that did not behave as intended, or communicating the feel of an application to other team members.

MI: *Designers have to make sure to always show some kind of feedback when the program's doing something.*

Flow. Designers need to create, iterate on, and communicate the *flow* of data, events, and other resources through the application they are designing (dataflow, event flow, workflow, timelines). Flow involves designing the structure of an application using a diagrammatic representation. Terms used by designers to refer to this view included: schematics, information architecture, wireframes, Visio documents, tree diagrams, timelines, etc. As shown by the strong link between *feel* and *flow* in Figure 1, one common way of representing feel is in a flow diagram.

FI: *I usually use PowerPoint by creating hyperlinks between slides, but there's no way to tell what slides connect to which other slides. I would love a way to visualize storyboard trees like that.*

Training. Designers need *training* resources (classes, online training, etc.) and other help. All of the

emails and forum entries analyzed were coded as training since they asked for help with a particular problem. Training topics ranged across a variety of issues including bugs in the code, external settings, and asking about courses on particular software. Sometimes, training was needed to overcome a selection barrier [9]:

> MI: *Sometimes you don't know if it's a lack in your own knowledge, or something the software can't do. […] It costs knowledge and time to figure it out.*

Reuse. One way in which designers first create their designs, or inspire themselves, is through reuse. Designers need tools to help support *reuse* their own (or others') code, image files, PowerPoint presentations, look, feel, etc., in creating a design.

> MI: *Snippet libraries to build your own code by taking code as is and using it as a Lego piece would be nice.*

Designers also need to sometimes recreate a look or feel seen elsewhere, such as "recreate the flash effect on this banner" [UO].

Themes. One need related to *reuse*, though not always overlapping with it, was *themes*. When themes and reuse overlapped, designers wanted to apply previously-created themes to their own designs.

> FI: *Just like PowerPoint has deck templates, Blend should have application templates (Office, web, etc.).*

Other times, creating a theme from their designs was in expectation of reuse.

Extensibility. Designers need to *extend* (and customize) software components, behaviors, and capabilities for their own needs and preferences.

> FI: *I would like better text control for making new fonts. Some software uses this squishing thing, rather than ultra-condensed weight.*

Sometimes, extensibility overlapped with themes and reuse. One designer wanted to extend her software's functionality by reusing someone's code for testing different WPF themes and skins [FO].

4.3. Designers' Iteration Needs

Once the initial version of one or more designs has been created, designers iterate on them. Since design verification often sparked new iterations, the codes we address in this section are: *usability* (and its neighbors *external constraints* and *granularity*) and *testing* (and its neighbors *automation, optimization, bugs, settings,* and *compatibility*).

Usability. Designers need tool support for evaluating *usability* issues in their design. These evaluations often result in further design alterations (e.g., "creating multiple prototypes and have the users pick the one they like best" [FO]). Another way of evaluating the usability of a design was by seeing how well it followed standard design rules.

> MI: *A tool would be nice where all of the usability rules you needed to follow were given in a table; to make sure you haven't overlooked any of them.*

As this last quote shows, usability sometimes overlaps with fixed external constraints.

External Constraints. Designers need to *keep track of external constraints* that limit the design: specifications for the design, the schedule, the size of the files, the performance of the application, additional software that needs to be downloaded before the application can be run, and the memory footprint of the application that is being designed. With design, some of the constraints are highly subjective, but the software could give recommended outputs.

> FI: *The design has to meet certain guidelines (spacing around images, contrast ratio, location, etc.). Maybe software could help make sure that these are met.*

Granularity. To more easily evaluate the usability of their prototypes, designers need to easily switch *levels of granularity*: moving from high-level schematics, to a detailed view of each screenshot, to the code behind the application, to the preview of the application, and back.

> MI: *I do a lot of switches like that (high level to lower levels) to find and fix errors.*

Testing. *Testing*, in its purest form, means to evaluate whether the design gives the correct output for particular inputs. Testing is an important need since interactive application designs are often highly complex and data-dependent.

> MI: *Tools should provide a mock service to generate data to hook up to the designed application. It's very hard to design a data-centric application without data.*

As Figure 1 shows, the two most highly related needs were *testing* and *flow*: seeing how an output resulted from the flow of elements through the design. This is a concept similar to slicing, which has been found to be useful in end-user programming environments (e.g., [14]). Such tools should not be limited to code, but also lower-fidelity prototypes.

Automation. Designers need *automation* tools in order to quickly perform one action on many different parts of the design, or to otherwise automate (record and replay) redundant steps of the design process.

> MI: *If the computer could create a style-guide-in-a-box specification template, you could just define what you'd like your specifications to look like, and the software would spit them out for a designed application.*

Optimization. Some designers wanted automation capabilities to quickly *optimize* the look or feel of the design. Optimization did not come up often, but was reported to be a very hard problem to solve. In perfecting visual aspects of the application:

> MI: *One way of quickly visualizing the best option is to generate sample previews of what the image would look like with the different number of colors.*

Another commented about optimizing the user's interaction with the application:

> MI: *Workflows with fairly detailed screenshots are useful for making decisions about unnecessary pages.*

Bugs. Designers need *debugging* tools to help them fix bugs. Bugs addressed by the users were sometimes specific errors in their code, and sometimes more general pain points. Some of the bugs were *usability* bugs, while others were just *incompatible* software.

> MI: *Things like case-sensitivity and overwrites are tough bugs to avoid and fix.*

Settings. Incorrect output was sometimes not caused by a bug, but instead by incorrect settings on the designers' computer. These two instances were hard for designers to discern. Even when they were distinguishable, figuring out the wrong setting or missing software to download was a daunting task.

> MO: *Images that are posted live online appear as broken images in the design mode. [...] I've seen this crop up a few times on these boards, but still can't seem to figure out what settings I'm missing on my end.*

Compatibility. Problems related to settings are exacerbated by the amount of switching designers do between different environments in both creating their designs (e.g., Visual Studio and Blend), and predicting environments used by the consumers (e.g. Internet Explorer and Firefox). *Software compatibility* tools are needed to help when a design is working in one environment, but not in another.

> MI: *It would be nice to render one web page in three different browsers with one click.*

4.4. Designers' Communication Needs

Designs are boundary objects, shared by designers, developers, users, testers, and usability researchers, among others. It is therefore not surprising that many have considered *communication* the ultimate goal of design (e.g., [10]). *Jargon* and *clean code* are related.

Communication. Designers need *communication* tools to help them report their detailed vision of the design to developers, program managers, and others. Without them, the wrong look or feel can be introduced into the design by those who create the working code. As we mentioned earlier, a flow diagram was often used to design the application's interaction. It was also the representation of choice for communicating that interaction,

> FI: *Another thing that the diagram view helps with is that sometimes developers won't see the forest for the trees, and will focus on the details. If the interface is instead explored at a step above the screenshots, then the presentation and the design meeting become more straightforward and focused on the interaction.*

It is also important to provide communication from developers back to designers.

> FI: *Designers should sometimes be given guidance on what controls to use by the developers.*

Jargon. Unfortunately, when interacting with developers and with code, designers need to understand or somehow translate *developer jargon* (written or verbal) they encounter from discussions, error messages, warnings, etc. While sometimes error messages are useful, other times, they are not:

> UO: ***Error** Scene=Scene 1, layer=Actions, frame=3:Line 3: Syntax error. function () {*

Cleanup. Some tools automatically generate code; however, in those situations, designers need *code cleanup* tools to remove unused (filler) code.

> *When generating code, a lot of 'junk code' is produced that developers complain about. If there is an error in the code, the designer then needs to do a code review and strip the unnecessary parts, which is hard!*

5. Validating Needs: A Focus on Flow

To validate our grounded theory work (the 20 needs presented in the previous section), we conducted an interview with two senior designers and two focus group sessions with nine designers. Our goals during those sessions were to: (1) get designers' opinions on whether the needs we found are indeed real to their work, and (2) gauge the relative importance of each need for the creation of tools to support designers.

The first validation session was a two-hour unstructured interview with two senior designers (one female). Both designers received a lunch voucher for their participation. These designers were managers, so their anecdotes about designer needs not only spanned their own experiences, but also those of less experienced designers on their team. When asked to describe the top five needs that should be supported through software, one of the designers yelled "Flow!" and the other agreed. Having a hard time narrowing the list down to five needs, they mentioned their top eight: flow, training, testing, reuse, communication, usability, external constraints, and optimization.

The second and third validation sessions were each a two-hour lunch-time focus group session. Pizza was provided for the session and each participant received a lunch voucher. Nine designers (two females), from eight product teams, attended the sessions. The types of designers included: interaction designers, graphic designers, web designers, design lead/managers, systems designers, and motions designers.

Focus group participants were first asked to go through the list of 20 needs individually and mark the needs which they had personally encountered in their design work. Table 2 shows the number of designers who experienced each need (one participant forgot to answer this part of the question). Even with this small sample, we can see that every need was experienced by at least one designer and 12 of the needs were experienced by more than half of the designers.

The participants were then asked to rank the top five most important needs in their own work. As Figure 2 shows, flow and feel were mentioned as being the most important needs. Flow involves creating, iterating on, and communicating the higher level structure of the application. Feel involves creating, iterating on, and communicating the users' interaction with the application. Additionally, all of the designers ranked flow as being one of the top five needs.

Both interview and focus group sessions validated the needs we derived through our grounded theory work as indeed being needs that designers encounter in their everyday work. Furthermore, even with this small number of participants, clear patterns emerged: (1) Each of the eight top needs the interviewed designers wanted tool support for was a top-five need for at least one of the nine focus group designers, and (2) Flow was identified as being the top need from both the interview and the focus group sessions.

6. From Needs to the Design of Tools

The list of 20 needs, and each need's tie to other needs, acted as a useful discussion facilitator which led to the design of several unique software tools. Designers' initial survey, email, forum, and interview tool requests were often incremental improvements to existing software. However, during the focus group sessions, several completely new designs for designer tools arose. Two examples are briefly described here.

(1) A "design-centric versioning system" that allows designers to *1*-keep track of alternate latest design versions, *2*-show the hierarchical ties between designs, and *3*-keep a list of design decisions as a part of the versioning system (e.g., attaching the scenario which led to the decision).

Table 2. Number of designers who reported experiencing each need.

Flow	8	Propose	6	External Constraints	4
Look	8	Communication	6		
Feel	7	Reuse	6	Automation	3
Optimize	7	Extensibility	5	Settings	2
Testing	7	Compatibility	5	Jargon	2
Themes	7	Granularity	4	Cleanup	1
Usability	6	Training	4	Bugs	1

(2) A "multi-view software development environment based on team members' role" to provide all software professionals a common artifact to work on, where each view would be a different facet of the software, directly connected to all the others.

Thus, the results presented here can be used to help in brainstorming about new design tools in addition to incremental improvements to existing software.

7. Discussion and Conclusion

This paper presents grounded theory findings from two studies to determine interactive application

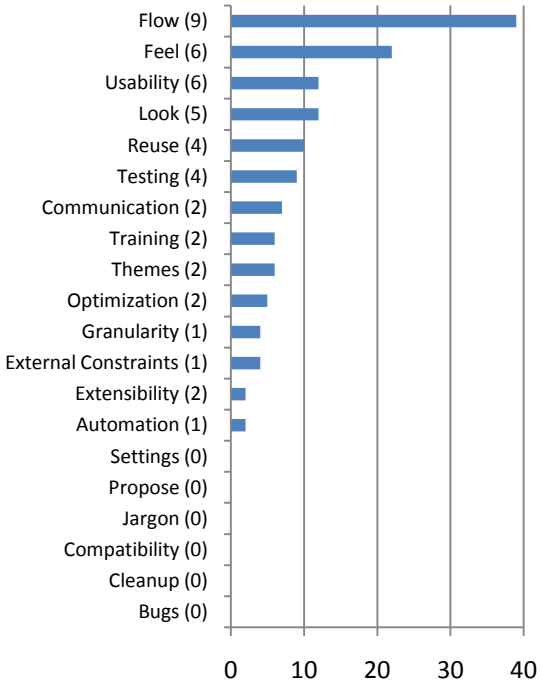

Figure 2. Importance rating (x-axis) for each need (y-axis). Ratings were calculated by summing the ranks assigned by each designer (rank 1: 5 points, 2: 4 points, etc.). The number in parentheses indicates how many designers ranked this need in the top 5.

designers' needs. A content analysis of 282 designers' artifacts identified 20 design creation, iteration, and communication needs (in order of importance): flow, feel, look, usability, reuse, testing, communication, themes, training, optimization, external factors, granularity, automation, extensibility, bugs, cleanup, compatibility, jargon, propose, and settings.

Follow-up interview and focus group sessions validated these needs and further explored the importance of each. All of the needs were validated by at least one of the focus group participants, with flow and feel being the most commonly reported. Flow was ranked as being the most important need.

Our data collection and analysis processes were strongly triangulated, to reduce bias. The fruits of triangulating were evident. For example, collecting artifacts only from online sources (emails and forums) would have biased our findings toward "training" and "bugs," since the majority of those artifacts contained a question about why their code did not work. However, collecting data from several other sources (such as the survey and interviews), helped identify and avoid this bias. A possible limitation of our study was that the validations were based on Microsoft employees' feedback alone, and therefore may not generalize to other designer populations. However, our population was diverse within the company, spanning several types of designers on thirteen different product teams.

The most surprising results of this work are the number of needs expressed by designers, their ties to each other, and their low software support. Based on these results, it is clear that interactive application design environments should provide explicit support for the 20 needs presented in this paper, and especially for *flow*. Recent HCI work on trajectories, such as the conceptual framework for trajectories presented in [2] is likely to be useful in the design of tools for supporting flow in designers' software.

Each of the needs reported here is a starting point for designing one or more features of a future design tool. Carefully designing a set of features to meet these needs will require a deep understating of how the needs overlap and interact with each other. This has strong implications for future work. Figure 1 can be used in framing the structure of future studies about the needs and the design of software to support them. Two novel designer tool ideas were already presented here as a result of employing this approach.

7. Acknowledgements

We thank our studies' participants, Gina Venolia for help recruiting participants, and, Margaret Burnett, Kael Rowan, and the reviewers for thoughtful input.

8. References

[1] Ambler, A. and J. Leopold, "Public Programming in a Web World", *Visual Languages*, Nova Scotia, Canada, 1998.

[2] Benford, S., Giannachi, G., Koleva, B., and Rodden, T. 2009. "From Interaction to Trajectories: Designing Coherent Journeys through User Experiences," *Proc. CHI 2009*, ACM Press, 2009, pp. 709-718.

[3] Burnett, M., S. K. Chekka, R. Pandey, "FAR: An End user Language to Support Cottage E-Services", *IEEE Symposia on Human-Centric Computing Languages and Environments*, Stresa, Italy, 2001.

[4] Chandrasekaran, B., Design Problem Solving: A Task Analysis, *Artificial Intelligence Magazine*, 11:59–71, 1990.

[5] Charmaz, K. "Grounded Theory: Objectivist and Constructivist Methods," pp. 509-35 in *Handbook of Qualitative Research*, 2nd ed., edited by N. K. Denzin and Y. S. Lincoln. Thousand Oaks, CA: Sage, 2000.

[6] designer. (n.d.), WordNet® 3.0, retrieved from http://dictionary.reference.com/browse/designer, December 04, 2008.

[7] Glaser, B.G. and A.L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Chicago: Aldine, 1967.

[8] Glaser, B. G. *Theoretical Sensitivity.* Mill Valley, CA: Sociology Press, 1978.

[9] Ko, A. J. Myers, B. A., and Aung, H. "Six Learning Barriers in End-User Programming Systems", *Proc. VL/HCC 2004*, pp. 199-206.

[10] Myers, B., Park, S., Nakano, Y., Mueller, G., and Ko, A, "How Designers Design and Program Interactive Behaviors", *Proc. VL/HCC 2008*, pp. 177-184.

[11] Newman, M. and Landay, J., "Sitemaps, storyboards, and specifications: a sketch of Web site design practice", *Proceedings of the 3rd Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques*, ACM Press, New York, NY, 2000, pp. 263-274.

[12] Newman, M., Lin, J., Hong, J., Landay, J., "DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice", *Human-Computer Interaction*, 2003, pp. 259-324.

[13] Rode, J., Bhardwaj, Y., Prez-Quiones, M., Rosson, M., Howarth, J., "As easy as "Click": End-user web engineering", *International Conference on Web Engineering 2005 - Lecture Notes in Computer Science 3579*, Berlin: Springer-Verlag, 2005, pp. 478-488.

[14] Rothermel, G., Burnett, M., Li, L., DuPuis, C., and Sheretov, A., "A Methodology for Testing Spreadsheets", *Transactions on Software Engineering and Methodology*, ACM Press, 2001, 10(1), pp. 110-147.

[15] Smith, G., and G. Browne, "Conceptual foundations of design problem solving," *IEEE Transactions on Systems, Man and Cybernetics*, 23(5), 1993, pp. 1209-1219.

[16] Wing, J., "Computational Thinking", *Communications of the ACM*, March 2006, 39(3), 33-35.

[17] Wolber, D., Su, Y., and Chiang, Y., "Designing Dynamic Web Pages and Persistence in the WYSIWYG Interface", *Proc. IUI*'02, 2002, pp. 228-229.