End-User Service Composition in Ubiquitous Computing Environments

Mark Webster Newman

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-138
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-138.html

November 27, 2007

End-User Service Composition in Ubiquitous Computing Environments

by

Mark Webster Newman

B.A. (Macalester College) 1992
M.S. (University of California, Berkeley) 2000

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the
GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor James Anthony Landay, Co-chair
Professor John Canny, Co-chair
Professor Athony D. Joseph
Professor Peter Lyman

Fall 2007

The dissertation of Mark Webster Newman is approved:

Co-chair Date

Co-chair Date

Date

Date

University of California, Berkeley

Fall 2007

End-User Service Composition in Ubiquitous Computing Environments

by

Mark Webster Newman

ABSTRACT

End-User Service Composition in

Ubiquitous Computing Environments

by

Mark Webster Newman

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor James A. Landay, Co-chair

Professor John Canny, Co-chair

The era of ubiquitous computing is upon us. We are seeing a sustained explosion in the numbers and types of networked devices and services with which users can interact. This means that great new capabilities are available to end-users, but such capabilities may come at a cost in terms of the complexity of understanding and managing multiple heterogeneous devices and services. In this dissertation, I present work on the design, development, and evaluation of three systems that offer solutions to existing approaches' shortcomings with regard to developing networked devices and services for end-users.

The Obje Framework is a distributed middleware platform that overcomes the problem of *piecemeal interoperability* by providing a robust interoperability solution for distributed services by dictating minimal up-front agreements and allowing the details of interoperation to be supplied at runtime through the delivery of mobile code.

The Obje Display Mirror (ODM) is a service that allows users to connect their laptops to any shared display device within a collaborative work environment, enabling seamless access to and interaction with remote devices. A study of ODM usage across six months indicated that its adoption had impacted workplace information sharing practices in a positive way.

OSCAR is an application that allows users to discover, control, and connect devices and services in a home media network. It leverages Obje to provide solutions to both *piecemeal interaction* and *sluggish adaptation* by allowing integrated control of all devices on the home network and allowing end-users to compose their own functionality from disparate devices and services. A two-phase user study involving 18 participants with varying degrees of technical skill demonstrates that users could employ OSCAR to create and access a range of functionality and that users were able to identify a wide variety of needs for which OSCAR would provide assistance.

The experiences with these systems reported in this dissertation point towards principles for designing frameworks and end-user tools to support an integrated, yet flexible and customizable user experience of ubiquitous computing environments.

____
Professor James A. Landay
Dissertation Committee Co-chair

____
Professor John Canny
Dissertation Committee Co-chair

To my amazing long-term domestic partner,
Valerie Jo Taylor

Table of Contents

List of Figures

List of Tables

Acknowledgements

It takes a village to raise a dissertation. I had an enormous amount of support both professionally and emotionally without which this document would have never come into existence.

My primary advisor, James Landay, refused to listen to me the several times I told him I was abandoning the Ph.D., and waited patiently each time until I came to my senses. His confidence in me (which greatly exceeded my own at many points) is probably the primary reason that I was eventually able to bring this project to a successful conclusion. John Canny stepped in to help advise me after things became complicated with James' departure to U. Washington, and his assistance was invaluable. I am deeply indebted to the now-departed Peter Lyman whose early enthusiasm about my project played a key role in motivating me to finish.

The work reported in this dissertation was largely carried out while I was employed at the (Xerox) Palo Alto Research Center. While I was there I had the privilege of working with an amazing set of collaborators. Several of those collaborators worked with me directly on aspects of the work in this dissertation. Keith Edwards was the originator of the Obje (neé Speakeasy, a.k.a. RNC, a.k.a. Recombinet) project and the mastermind behind the concept of recombinant computing that underpins the Obje Framework. It was a great pleasure and an enormously educational experience to work with Keith during the four years we collaborated. His vision and perspective is inextricably intertwined with all of the major themes of this dissertation. I had the pleasure of working shoulder-to-shoulder with Trevor Smith for nearly the entire seven years I was at PARC, including the many, many months of pair programming which were probably the most stimulating periods of my career so far. Trevor's challenging questions and insights played an instrumental role in shaping all aspects of this dissertation work, and in shaping my views about how to Get Things Done in general. In addition, his technical craftsmanship is unparalleled in my experience and serves as a standard for producing quality software that I shall probably never meet. Jana Sedivy was the fourth member of the original Four Horsemen of Obje, and was a great pleasure to work with. She played the critical role of keeping the project somewhat sane and keeping the rest of us from drifting off into intellectual irrelevance or incessant "code-rewhacking." Chronic intern Shahram Izadi was a shot of adrenaline and fresh ideas when he was in town, and was always a joy to work with.

Nic Ducheneaut was my key collaborator on the Obje Display Mirror project, and his influence was significant, especially in deciding how to conduct user observations. As my next-door office neighbor and fellow Golden Bear, he was also the first place I would go to gripe about PARC management and UC Berkeley bureaucracy. In the last few years, he was also the first place I would go to talk through ideas and plans to get his feedback on what made sense to say and do. I worked intensively with Ame Elliott on the OSCAR user studies, which was an absolute pleasure, and especially satisfying after trying to find a way to collaborate with her since the early days of GUIR in 1997-1998. Ame and Nic were also very important sources of encouragement and support during the final throes of trying to finish the dissertation work and get on with my life. Beki Grinter and Allison Woodruff, along with Nic and Keith, rounded out the "Ass Kicking Committee" that I convened to help keep me focused on the dissertation amongst all of PARC's distractions. Their feedback and encouragement was quite helpful, though frankly it turns out I could have stood to have my ass kicked a bit more. Still could.

Finally, Bo Begole and Kurt Partridge came into the PARC picture too late to be direct collaborators on the work presented here, though my other collaborations and interactions with them certainly made my last couple of years at PARC stimulating and enjoyable. Bo was extremely supportive of me as my manager, and went above and beyond several times to ensure that I could make progress towards finishing my dissertation, particularly in the face of bizarre and challenging roadblocks erected by a certain governmental agency whose name rhymes with "pissed."

Scott Klemmer was a tremendously valuable resource as I laboriously progressed through the various stages of dissertating. Though he came to Berkeley as my “junior colleague” two years after me, he quickly surpassed me at which point I began to employ him as a scout in uncharted territory. His advice and guidance were instrumental in getting me through the final few hurdles and especially in navigating the perils of the academic job market. Jason Hong was also a good friend and confidant in these final phases (not to mention the very early phases when I was his Soda Hall officemate).

Jimmy Lin’s dissertation nightmare provided important lessons (thanks for taking that bullet), and his guidance through the writing process was helpful, not to mention his sharing of the very MS Word template within which I am writing these words. Scott Carter also provided key late-stage help with navigating the UC Berkeley bureaucracy and with maintaining contact with difficult-to-reach committee members (along with help from Nathan Good).

My mother never flagged in her support of me, whether the plan of the moment was to dedicate myself to finishing the Ph.D. or dropping out to start an organic goat dairy. I suspect she preferred the former, but did a pretty good job of keeping her opinions to herself. My father remained supportive throughout as well, despite my suspicion that he doesn’t quite understand why I don’t just go make a bunch of money like a normal person. I’m not sure I understand it either.

This dissertation lasted long enough to see the entire cycle of life play out in my family. My dear stepfather Robert Weiskopf passed away a few months before the completion of the dissertation. As he was a psychology professor and thus a representative of the “academic wing” of my family, along with my dear aunt and constant inspiration Ann Steiner, I would have loved to have him live to see me finish the project. At the other end of the cycle, Søren Francis and Ingmar Samuel Newman-Taylor showed up while the dissertation was still in progress, and while I can’t honestly say they “helped” the dissertation in any pragmatic way (in fact, strong arguments could be made in the opposite direction), they certainly did provide a strong motivation to finish the thing and get on with the next phase of life.

Finally, and by far most importantly, none of this would have been possible without the undying support, encouragement, and confidence of my amazing and wonderful long-term domestic partner (archaically referred to as “my wife”) Valerie Jo Taylor. Her constant support during my many setbacks, changes of heart, and emotional collapses was the glue that held the contraption together long enough for it to putter over the finish line. It is to Valerie that I dedicate this dissertation and my love.

1 Introduction

The era of ubiquitous computing [160] is upon us. We are seeing a sustained explosion in the numbers and types of networked devices and services with which users can interact. At the same time we are constantly expanding the environments in which users want or expect to be able to take advantage of computation. Implicit in this is the fact that the number of tasks or activities that users can carry out using computers is constantly increasing. In the leisure domain, activities such as watching TV and listening to music are increasingly carried out using general-purpose computers or using special-purpose devices with significant general-purpose communication and processing capabilities (e.g., TiVos, iPods), which open up vast new possibilities in terms of the patterns and practices of media consumption. In the workplace, computing technology has long been dominant, yet the variety of activities to which it is applied is constantly growing as patterns of communication and data sharing emerge along with new patterns of corporate organization and collaboration. Devices in the workplace are more easily networked, as well, allowing for the incorporation of various media consuming-

and producing-devices into work practices (e.g., data presentation, videoconferencing).

The explosion of networked devices and services is being facilitated by the increasing availability of data networks in a variety of environments, including the home, the workplace, and public spaces. It is further being facilitated by the development and adoption of interoperability standards for networked services such as Universal Plug and Play (UPnP) [153] and Web Services standards such as the Web Services Description Language (WSDL) [23] and the Simple Object Access Protocol (SOAP) [58]. These developments lay the basic groundwork so that, in principle, almost any device can interact with any other device. However, these approaches alone do not guarantee that users will be able to connect and control the devices and services they need to use in order to carry out the tasks they want to accomplish.

1.1 Problems and Opportunities with Networked Services

From a user's perspective, the explosion of networked devices and services has the potential to transform the experience of interacting with many aspects of the surrounding environment. On the one hand, awesome new possibilities emerge from the ability to interact with devices, information, and other people both locally and at a distance, as well as from the ability to combine functionality from multiple sources to effect new applications. On the other hand, the realization of these possibilities will not come about of their own accord, but will rather require new approaches to interoperability, interaction design, and application development. In this section I will present an ideal scenario of near-future home network use, along with a discussion of the challenges that must be overcome to realize such a scenario—in particular the challenges that will face users as they attempt to make sense of the capabilities of their environments and the tools at their disposal for accessing those capabilities. Following the discussion of problems, I will outline the coordinated approach to delivering a sensible, integrated user experience explored in this dissertation—an approach that provides solutions for robust interoperability, integrated multi-device control and interaction, and support for lightweight end-user composition of varied devices and services. In the final section of this Introduction, I will present the central thesis of this dissertation and outline the concrete contributions that my work has made in support of this thesis.

1.1.1 Near-Future Home Networking Scenario

The following scenario illustrates both the benefits and challenges of near-future home media networks.

> *Alice is sitting in her living room on a Saturday afternoon reading news on her laptop. She decides she would like to listen to some music so she navigates to her favorite online music service and arranges to have some music streamed directly to her*

> *living room speakers. A short while later, the doorbell rings. Engrossed in the news, she does not wish to get up, so she directs the webcam above the front door to send its output to a digital picture frame that is mounted on the wall next to her. Seeing a delivery person on the screen, she remembers that she is expecting a package, so she connects her laptop's microphone to the intercom speaker next to the front door to say "Hold on, I'll be right there." The delivery person's "OK" is heard through the living room speakers as Alice leaps off the couch to head for the door.*

The experience presented in this scenario is one of convenient, seamless access to a variety of resources both within and without the home. The benefit to Alice is that she is able to connect and control her available devices and services with little effort and minimal advance planning. The challenges to realizing such an integrated user experience, however, are substantial.

The first challenge is that there needs to be a plethora of networked devices, and that currently hardwired devices such as speakers, microphones, and displays need to be recast as networked services that can be easily connected and disconnected from each other. This challenge is being addressed quite rapidly by the consumer electronics industry as evidenced by the rapid proliferation of new networked media devices that are announced at an ever-increasing rate [44]. However, the mere existence of such devices is not sufficient to enable the scenario described. Current approaches to the design of networked services and applications suffer from two key problems: *piecemeal interoperability, piecemeal interaction, and sluggish adaptation*.

Piecemeal interoperability refers to the situation where a service is designed to interoperate with a particular subset of its peers but not with others. For example, it is not hard to imagine buying a set of networked speakers that are designed to work well with one or more streaming music services but would be difficult if not impossible to use with an in-home intercom system.

Piecemeal interaction describes the experience of using different applications, often with very different user interface styles, to interact with and control the different devices and services with which one interacts. In the scenario above, this would occur if Alice needed to use one application to control her streaming music and a different one to connect the webcam to the picture frame. The fact that different applications may be required for these different operations is less critical than the fact that these applications may have significantly different user interfaces and require Alice to learn and master different interaction styles. In the worst case, these applications may require Alice to interact with different physical devices in addition to employing different techniques. Again, based on existing networked services and the applications to control them, it is easier to imagine a world of piecemeal interaction than one that supports the integrated user experience posited by the above scenario.

Sluggish adaptation is the problem that is experienced when capabilities that are theoretically available within a collection of devices fail to be realized by users because the application user interfaces available to them do not expose them. In the domain of networked services, this problem is dramatically exacerbated when the capabilities in question are emergent properties of the combinations of devices available in any one user's particular environment. As such environments increase in size, variety, and number, and as the amount of and variety of functionality desired by users increases, traditional developer-based approaches to application development will prove to be unacceptably slow. The scenario above would be sabotaged by sluggish adaptation if Alice were unable to connect the front door webcam to the picture frame in her living room, despite the fact that both are networked devices that share a common video data format, or if Alice does not have access to a client application or device that provides the ability to connect these particular devices together. Perhaps the developers of the webcam had only anticipated that users would want to access its stream from a web browser running on a conventional laptop or desktop computer, and the picture frame developers only anticipated that their device would be connected with a user's digital family photo and video albums. Even after a set of users recognize that a webcam-to-picture frame connection capability would be valuable, conventional approaches require that, in almost all cases, someone with software development skills invest time and effort into creating the needed application functionality before users can reap the benefits of the capabilities that, in some sense, they already possessed.

1.2 Towards an Integrated User Experience

Given the anticipated continued churn in devices, services, media formats, and data transfer protocols, it will be practically impossible for application developers to provide custom-built applications that support the range of things that people want to do given the variety of resources they have available at any particular point in time. In fact, "applications," by their very nature, are piecemeal. They are designed to run on particular platforms and devices, they are designed to interact with specific types of devices and services, and they are designed to allow users to carry out specific tasks. A promising approach to overcoming these various issues and to providing an integrated overall user experience, which is the approach explored in this dissertation, is to provide basic tools and infrastructures that allow users to compose resources flexibly and intuitively to carry out the tasks they want to accomplish. Indeed, it is end-users themselves who are in the best position to know not just *what* they want to do but also *which* resources they have available and *how* those resources should be composed. Of course, they may not know all of the details of *how* to assemble the given resources—for example the best protocol to use for data transfer, or the specific parameters that need to be set to use a particular service, and this is where my research comes in. A central goal of the research in this dissertation is to find the sweet spot on a spectrum that ranges from extreme ease of use for any particular application—represented by pre-built single-function applications or devices—to extreme flexibility in accomplishing a variety of tasks—represented by general-purpose programming languages like Java or C++. I refer to the sweet spot being sought as "end-user composition" to express both its affinity with and difference from the more widely explored concept of "end-user programming," which broadly speaking is intended to allow end-users to express very rich and in many cases arbitrary programs using metaphors and tools that are easier to learn and master than general purpose textual languages. I define an *end-user composition* system as a system that provides end-users with *coarse building blocks* that can be assembled in *simple, constrained ways* to accomplish a *variety of tasks.* End-user composition is aimed at a point in the ease-of-use/flexibility spectrum that is less flexible and powerful than end-user programming, but ideally easier to use for untrained users (see Figure 1-1).

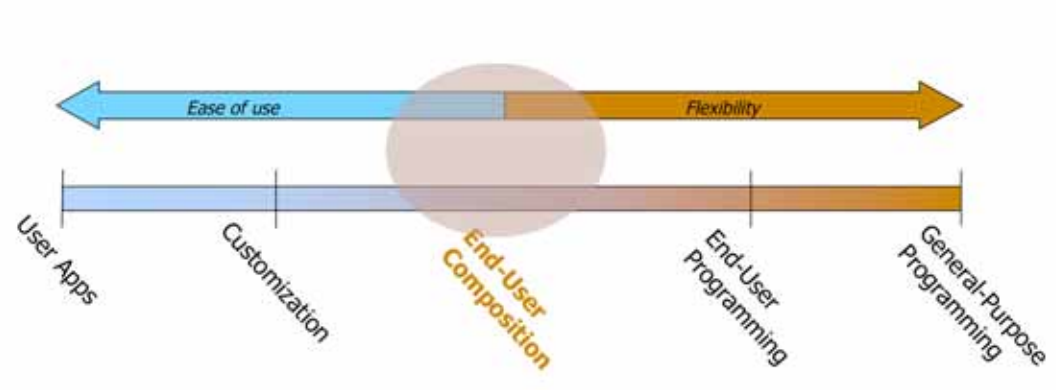


Figure 1-1: Functionality can be delivered to users in ways that span a spectrum between ease-of-use for any given task and the flexibility to accomplish various different tasks. This dissertation explores end-user composition, which is an attempt to locate an optimal tradeoff between ease-of-use and flexibility in the domain of networked services.

Supporting an integrated user experience of networked devices and media requires research at two levels:

- Infrastructure: How should the constituents of the network be built so that they can be discovered, composed, and controlled by end users?
- End-user tools: How can we build tools that allow untrained, non-technical end-users to compose and control complex networks of devices to accomplish a variety of tasks?

The work presented in this dissertation addresses the key questions at both of these levels. At the infrastructure level, I will present the Obje Framework, which is an interoperability framework for networked services that supports robust interoperability among devices and services that have little prior knowledge of each other's inner workings and also supports end-user composition[1]. At the level of end-user tools, I will present two systems along with their evaluations: the Obje Display Mirror and OSCAR.

The Obje Display Mirror, allows users to easily discover public displays in a workplace setting and mirror their laptop's display onto them. It leverages the Obje Framework to enable users to perform an otherwise tedious task, and provides advantages over the VGA cable-based display connection system that it replaces. A year-long study of adoption and use of the Obje Display Mirror showed that its availability produced subtle but important shifts in users' patterns of behavior, and also illuminated some of the advantages and problems associated with re-casting a commonly used function from a tangible, hardware-based interaction into a virtualized, networked service-based interaction.

OSCAR provides a touch-based user interface that allows users to discover, connect, and control a variety of Obje services and devices on a home network, and also allows them to create *Setups*—reusable compositions of services to carry out recurring activities. A study involving 18 users with varied technical backgrounds revealed that users could effectively user OSCAR to create both ad-hoc and reusable compositions to accomplish varied tasks. The results of the study

[1] Obje also supports transport-layer security (TLS) to allow all communication among services to be authenticated and encrypted. This feature of Obje is not discussed further in this dissertation, but an demonstration of Obje that highlights its security features is described in [136].

also suggested guidelines for the development of future systems with similar goals as well as directions for future research.

1.3 Thesis and Contributions

The **thesis statement** of this dissertation is:

> *An integrated, yet flexible and customizable user experience of interacting with multiple heterogeneous devices and services is achievable through a combination of*
>
> - *a middleware framework that supports robust interoperability;*
> - *mechanisms and end-user tools that allow ad hoc connections among distributed devices and services;*
> - *framework- and application-level support for dynamically distributable control that allows users to monitor and control both individual devices and ongoing connections; and*
> - *end-user tools that support the discovery, connection, and control of individual devices as well as the creation, modification, and invocation of both temporary and reusable service compositions.*

It is not the goal of this dissertation to evaluate the relative contribution of each of these components to a compelling user experience of ubiquitous computing environments. Rather I wish to argue that all of these components are needed to achieve the end goal. My work provides a complete, coherent demonstration of an end-to-end system bringing together each of these components in order to deliver an integrated, flexible, and customizable user experience. Throughout this dissertation, I will provide support for the thesis statement by describing my work's three main contributions:

1) *A service framework that supports robust interoperability and end-user composition.*

   The Obje Framework dictates a minimal set of a priori agreements among cooperating services and employs mobile code to allow one service to extend another's behavior at runtime, thus lowering the bar for interoperability. In addition, Obje defines simple, standard service interfaces that describe the roles that services can play in compositions, thus laying the groundwork for end-user composition.

2) *A case study of a shared display service that demonstrates how a persistent, networked, user-accessible service can provide advantages over the hardwired legacy system it replaced.*

   The Obje Display Mirror lowers the barrier for displaying information to other co-located users and allows multiple simultaneous users to connect easily to public displays. Positive effects on user interactions (such as increased incidence of multi-user display use and verbal reports of improved satisfaction) were observed over an extended period of time. This contribution includes the application itself, including its novel features, the lessons learned from observing users, and the novel adaptation of existing observation methods (especially Lag Sequential Analysis [51]) employed to conduct comparative analyses of user behavior before and after the service was deployed.

3) *A novel application and user interface for end-user composition in home media networks that can be used effectively by people with a range of technical skill to accomplish a variety of tasks.*

The OSCAR application is an embodiment of an end-user composition system built atop a middleware framework that supports robust interoperability, ad hoc connections, dynamically distributable control, and support for both temporary and reusable compositions of devices and services. It thus provides a working, testable example of the "integrated, yet flexible and customizable user experience" described in the thesis statement. A two-phase user study showed that users could employ OSCAR effectively to accomplish a range of tasks, and that they preferred OSCAR to their current systems for accomplishing similar tasks.

1.4 Dissertation Outline

The remainder of this dissertation discusses relevant related work, describes in detail each of the contributions enumerated in the previous subsection, indicates directions for further research on this topic, and provides a set of conclusions.

In ***Chapter 2***, I present related work that has attempted to address one or more aspects of delivering an integrated user experience of ubiquitous computing, and describe how such work informs, complements, and differs from the research in this dissertation. In ***Chapter 3***, I describe the Obje Framework and show how its unique "recombinant computing" approach to network service design provides a robust and flexible solution for interoperability among arbitrary devices and services. The Obje Display Mirror application design, development, and evaluation is the topic of ***Chapter 4***. OSCAR is described in the following two chapters, with ***Chapter 5*** focusing on the design and implementation, and ***Chapter 6*** focusing on the user study and results. In ***Chapter 7***, I discuss pathways for future work that have the potential to build upon this dissertation to deliver an even more comprehensive integrated user experience of ubiquitous computing. Finally, in ***Chapter 8***, I conclude by returning to the problems, solutions, and contributions outlined in this Introduction and summarize the ways that the work described in Chapters 3-6 connects with the broader goals and claims of the dissertation as a whole.

2 Related Work

In this chapter I will present a discussion of related work that covers both infrastructures for combining networked services as well as end-user tools for effecting compositions. The unifying theme of all of the work presented here is that it plays a role in providing an integrated user experience of interacting with an environment of networked devices and services. However, as we shall see, there remain significant gaps in the literature and it has been my goal to address these gaps.

2.1 Overview of Related Work

On the infrastructure side, there are two main bodies of work to be considered: *Interoperability Frameworks* and *Ubicomp Application Frameworks*. The work on *Interoperability Frameworks* comprises initiatives from the software and consumer electronics industries that have attempted to define standards of interoperation among services in various domains. These are important because they demonstrate a strong desire on the part of industry to promote interoperability among devices and services produced by different manufacturers. Further, it is essential to discuss them because in examining their shortcomings we will see why it was necessary to build the Obje Framework. The work on *Ubicomp Application Frameworks* has largely been carried out within the HCI, ubicomp, and systems research communities. The reason for considering this work is that, although many of the frameworks that have been developed do not address the same concerns as Obje, they often do share the same high-level goal of providing users with a seamless experience of interacting with multiple devices and services.

As I shall discuss, however, both the Interoperability and Application Framework approaches suffer from the same fundamental drawback: a reliance on software developers to perform the integration of services and provide functionality to users in the form of targeted applications. As described in the introduction, my work attempts to eliminate this drawback by providing support for end-user composition. Other approaches to reducing or eliminating dependence on developers have been proposed as well, and so before discussing work related to end-user composition, I will present work related to the competing strategy of *Automatic Service Composition*.

Automatic Service Composition systems take a description of a desired capability or application and a set of available services and automatically generate a composition of those services that provide the capability. Among the various systems in this category, the descriptions of desired capabilities can take different forms and can be produced by different parties such as developers, local administrators, or the end-users themselves. Often, the automatic composition systems themselves are agnostic to who has created the description or the form in which it is expressed, and instead focus on creating an optimal composition from the given description.

Finally, I will discuss two approaches that allow end-users to have an integrated experience of using multiple devices and services without the intervention of developers, designers, or administrators. The first, *Universal Remote Control,* employs techniques to allow devices to publish descriptions of their functionality in such a way that a client device can be configured (with varying degrees of automation) to control each device, either one-at-a-time or in conjunction with the other available devices. The second approach, *End-User Composition of Networked Services,* allows users to create persistent or re-usable configurations of devices and services that go beyond simple control. For example, a user can set up a connection between two devices that allow the first device to stream media to the second device, or can configure a device to respond to another device's change in state. The OSCAR system and user study described in Chapters 5 and 6 of this dissertation falls into the category of End-User Composition of Networked Services, and in my discussion of prior work I will discuss the shortcomings of work in this area that OSCAR was designed to address.

2.2 Interoperability Frameworks

With respect to networked devices and services, an interoperability framework is a set of standards that describe how a new device or service must present itself to existing entities on the network so that it can be used in certain prescribed ways. More precisely, an interoperability standard will describe how services are named and identified, how they can be discovered, how their capabilities are provided for inspection by clients and other services, and what particular service description or application programming interface (API) elements they must include to engage in particular types of interoperation. For example, an interoperability standard might dictate how a new networked music player can discover and connect to a networked audio rendering device (e.g., set of speakers) and be controlled from a user's home computer. Similarly, an interoperability standard might dictate how a new web-based mapping service must present itself so that it can be accessed by third-party services that wish to plot geographic data onto a graphical map.

2.2.1 Universal Plug and Play and the Digital Living Network Alliance

The Universal Plug and Play (UPnP) set of standards [153] was developed by a consortium of software and consumer electronics vendors in the hopes of ensuring interoperability among devices that are co-located on a local area network. A wide array of standards has been produced to define the operation of various classes of devices such as printers, scanners, Internet gateway devices, and audiovisual devices. As of July 2007, these standards have begun to be adopted in some areas—most notably in the Internet gateway device product category—, and in other areas additional consortia have been assembled to further specify the details of how devices will interoperate. One of the most significant efforts along these lines is the Digital Living Network Alliance (DLNA). The DLNA is focused on maintaining, evolving, and promoting the UPnP Audiovisual standard (UPnP AV) [126], and has built an additional set of standards [34] on top of UPnP to dictate the details of interoperation left open by UPnP such as the media formats and data transport protocols that A/V devices must support in order to be compatible with the standard.

DLNA needed to come into existence in large part to fill in the gaps left by the UPnP standard. This is because UPnP exemplifies a class of interoperability technologies that fundamentally require agreement on an expanding coterie of standards to achieve compatibility. Devices and applications are coded against the specifics of particular "device profiles"—a specification of the standard operations supported by a given device type—as well as the specific family of protocols defined by UPnP. A device, such as a networked camera, coded to use a UPnP MediaServer must be recoded to use a UPnP Printer, and again for a UPnP

MediaRenderer. Further, these profiles themselves are subject to change, meaning that an application coded against a particular version of a particular device's profile may not work with other versions. This situation limits evolvability in digital networks—the presence of a new device type requires that all *other* devices on the network be updated in order to use it.

2.2.2 Jini Network Technology

Technologies such as Sun's Jini [158] provide a degree of "insulation" between the APIs that client applications are coded against and the actual protocols used to communicate with a remote service. In Jini, mobile code is used to provide a service-specific implementation of an interface that a client is written to use. At runtime, then, this implementation can augment the client's functionality to communicate with the service using a protocol not previously built into the client. Still, however, like the systems discussed previously, Jini requires that clients and services be written against each other's specific interfaces in order to use each other. Jini provides a platform for mobile code on top of which a set of recombinant interfaces could be developed, but does not itself address the challenge of providing such a set of interfaces.

2.2.3 Web Services

Web services are the currently dominant evolution of re-usable object oriented programming architectures [78], whose evolution has included RPC systems

[162], distributed components [53, 119], and component frameworks [103, 142]. Web service frameworks [71] are distinguished from their predecessors by the protocols they use for discovery and messaging, and the formats used for service description, and not in any particular technological capability. Web services do bring something fundamentally new to the table, however—the notion of persistent and universally available services that can be used as components in a variety of systems. Even previous distributed component frameworks (e.g., [103, 106, 142]) did not tend to permeate beyond the enterprise network.

2.2.4 Shortcomings of Existing Interoperability Frameworks

Any of the approaches described in this subsection can serve as the basis for a framework that supports interoperable services and an integrated user experience. However, the current state of these efforts remains inadequate for a number of reasons. Firstly, each of them requires developer intervention to create new application functionality for users. Secondly, they stop short of defining sufficiently detailed service description standards to actually guarantee interoperability. This second point is at least partly overcome in the case of DLNA, which was formed in large part to address this very concern. They have extended UPnP to specify not only the types of services that can interact with each other, but critical parameters of interoperation such as acceptable media types and data transport formats. However, as I shall discuss in Chapter 3, the level of detail of these specifications, while necessary for ensuring interoperation, will also serve as an impediment to achieving sustained compliance among the many vendors hoping to participate in the standard. The large number of detailed agreements required to guarantee interoperability will be expensive and difficult to implement for many vendors, and, perhaps most importantly, they will serve as a barrier to evolution. In order for a new device type, media type, protocol, etc. to be introduced into the system, an additional set of standards must be written and agreed upon by the many participants in the alliance—a lengthy and expensive process. The Obje Framework, presented in Chapter 3, overcomes these shortcomings by dictating agreements about only the most basic levels of communication and interoperation, and allowing the detailed agreements to be worked out at runtime using the distribution of mobile code.

2.3 Ubicomp Application Frameworks

A number of frameworks have been built to make it easier to develop certain types of ubiquitous computing applications and, by extension, provide a compelling experience of interacting with multi-device environments to end users. Some of these frameworks are focused on supporting particular aspects of ubicomp applications, such as providing contextual user interface adaptation or supporting capture and access applications. Others of these are designed to provide more general application support to a wide range of possible applications that are coordinated across multiple devices and services.

2.3.1 Toolkits for Building Specific Types of Ubicomp Application

The Context Toolkit, Context Fabric, and INCA are examples of Ubicomp Application Frameworks that support a particular style of ubicomp application or support a particular component of a more general application style. The Context Toolkit [32, 134] is aimed at making it easier for developers to incorporate contextual inputs such as a user's identity and location into an application. In doing so, it implicitly facilitates the creation of compositions of various low-level distributed services in a way that is invisible to the end user and perhaps even to the application developer, at least inasmuch as the contextual data is provided by sensors that are controlled by hosts distinct from the application's host. Context Fabric [65, 66] explicitly supports composition of distributed services, as its model is based on the assembly of distinct InfoSpaces. Context Fabric's distributed architecture provides a general, shared infrastructure to support the construction of varied context-aware applications while also providing users and system administrators with the ability to define enforceable privacy and information access policies independently of any particular application. INCA [149] also explicitly supports coordination of distributed services by allowing its

Capturer, *Accessor*, and *Storage* components to run on different hosts and be located by applications via the *Registry*.

With each of these frameworks,, the target audience is the community of developers, and one of the functions of the framework is to make it possible to hide the distributed nature of the component services from the user and to a large extent from the developers as well. For the intended applications, this level of encapsulation is probably appropriate—the developer who is using the Context Toolkit to build a context-aware In/Out Board application may not need to know whether the presence of the application's users within the building is sensed using RFID badges, face recognition software attached to the building's security cameras, or keyboard and mouse activity detectors attached to the users' desktop computers. Certainly, one hopes, it will not be important for users to be aware of this level of detail. For other applications, such as the creation of an ad-hoc video conference, it may be quite important for the users to specify exactly which camera, microphone, display, and speakers he or she wishes to use for the application. In this latter case, exposing the specific devices directly to the user and allowing them to assemble them as they see fit is the best way to provide them with the functionality they need. Frameworks such as the Context Toolkit, Context Fabric, and INCA that are aimed at encapsulating the distributed nature of services in order to ease the task of application development are not well suited to supporting applications that require that users are aware of the distinct identity of each service.

2.3.2 General-Purpose Ubicomp Application Frameworks

More general-purpose Ubicomp frameworks such as iROS, GAIA, and one.world have also placed an emphasis on supporting developers.

The iRoom Operating System (iROS) [47] provides ad hoc interoperation of loosely coupled services by allowing them to share data through tuplespaces [19]. GAIA [131] attempts to extend traditional operating system concepts, such as file systems, process management, and user sessions, to multi-device, multi-user "Active Spaces." Leveraging well-understood constructs allows developers to more easily understand how to create applications for these new types of environments and adapt existing application patterns to the domain of ubiquitous computing. The goal of one.world [54] is to provide a new programming model for application development that emphasizes three principles that are more pronounced in ubiquitous computing than in traditional single-OS or distributed system: *embrace contextual change*, *encourage ad-hoc [automatic] composition*, and *recognize sharing as the default*. Aura [138] allows abstract definitions of users' "tasks" (i.e., collections of resources and services related to a particular user goal such as "organize conference" or "write dissertation," along with status information about the most recent usage patterns of those services and resources) to be executed in a variety of mobile contexts by adapting themselves to the available services. Services are provided according to the operations they support (e.g., "editText" or "watchVideo") and provide a range of parameters (e.g., display window dimensions, current scroll position, spell checking enabled/disabled) that can be accessed and controlled during the task initiation process. It is not clear what degree of standardization regarding service types and parameters, in conjunction with the task definitions that must control them, would be required to make Aura tenable across the varied, heterogeneous environments they envision, but it would appear to be quite substantial.

Each of these systems allow developers to create applications that seamlessly adapt to the addition or removal of clients and services, run on heterogeneous platforms, and accommodate multiple simultaneous users. In principle, each of them could be extended to support the ability of end users to compose services; however this is not a capability that any of them possess in their published forms. It is not the goal of such platforms to provide a set of standardized service types that can be assembled by end-users in pre-defined ways, and therefore it remains the responsibility of developers to create the applications with which users will interact.

2.3.3 Shortcomings of General Ubicomp Frameworks

The purpose of discussing these various Ubicomp frameworks is not so much to compare them to Obje or point out their deficiencies, but to point out the range of purposes for which such frameworks are built. Many of the features provided by iROS, Gaia, one.world, and Aura (e.g., interprocess event propagation, process migration, single-process adaptation to changes in execution context) are not provided by Obje, and several of the features provided by Obje (e.g., robust interoperability through mobile code exchange and dynamically distributable user interfaces) are not provided by these frameworks. From the perspective of this dissertation, the most important fact is that none of the four Ubicomp frameworks discussed here are capable of supporting end user composition without significant further specification. They do not dictate the specific inter-compatible service interfaces that are required to enable end-user composition tools to be written. While such interfaces could, in all likelihood, be defined on top of one or more of these frameworks to enable an application such as OSCAR, is is not a part of these frameworks' contribution to describe how those interfaces should be defined, and what additional framework features should be supported, in order to support end-user composition. In other words, while OSCAR could have been built atop any of these frameworks with enough additional design and development, there would be no particular advantage or cost savings in doing so[2].

2.4 Automatic Service Composition

Various research projects have investigated how a multi-component configuration can be automatically composed from a more abstract specification [52, 91, 108]. For the most part, these projects have been agnostic to how the specification for a composition was generated, or they assume that a programmer created the specification. A shared assumption, however, is that the details of the composition are of no interest to the user and should be hidden.

2.4.1 Automatic Data Flow Path Creation

Ninja's Automatic Path Creation [52] and the closely-related Service Composition work at Stanford [67, 80, 81] take a data flow approach to service composition. Sequences of data transforming services are composed automatically or semi-automatically (with the help of developer-created scripts) to deliver data from one service to another service or client in a requested format. Care is taken to

[2] The Obje Display Mirror, on the other hand, could be built atop any framework that provides ad hoc discovery and a reliable way to establish a streaming connection using a predefined data format. Thus, any of the frameworks described in this section and to a large degree the Interoperability Frameworks described in the previous section would suffice for that particular application. The unique contributions of Obje are far better reflected in the OSCAR system and study, whereas the Obje Display Mirror project provides other insights and contributions as will be described in Chapter 4.

ensure that the data flow paths provide optimal quality-of-service. Most importantly, the paths are constructed automatically and the details are hidden from users and in many cases from the developers of the applications. While this is appropriate for many of the situations that are targeted by these systems, such as Internet-scale services that present a simple, unified interface to consumers of data, these approaches will not suffice for situations where users want or need to assemble components in new ways to accomplish tasks unforeseen by developers.

2.4.2 User-Initiated Automatic Service Composition

Other projects have attempted to provide a solution for semi-automatic service composition that reaches all the way out to the end user. For example, both Task Computing [98] and InterPlay [100] perform automatic service composition based on simple inputs that are, in at least some cases, generated by end users. In the case of Task Computing, the inputs are pairs of services and in the case of InterPlay, the inputs are structured “pseudo-sentences” that can be constructed with the aid of a wizard-like user interface. Both systems strive to hide the details of the composition from the user, which may result in a fluid user experience when it works, but will leave the user helpless to understand what went wrong and how to correct it when things do not go as planned. Surprisingly, neither of these systems has been subjected to studies with users to determine how well they will serve users’ needs under realistic circumstances.

2.4.3 Shortcomings of Automatic Service Composition

While it is generally true a large portion of the time that the details of how a computational system is going about accomplishing its task are of little or no interest to the user, it cannot be said that such details are never of interest. It is important for users to be able to understand and control their networks at some level because, ultimately, when engaging in setup or maintenance tasks, they will have to understand something about what is going on. For example, if an attempt to carry out an activity such as “listen to music in the kitchen” fails, it will fall to the user to attempt to understand if the failure occurred because the user’s action to initiate the activity was incorrect or misinterpreted, whether the system was unable to effect the composition because of an incompatibility, miscommunication, or failed operation, or whether some out-of-band problem has occurred such as a general network failure, wire disconnection, or hardware failure. Systems that take control away from users in the interest of making everyday interactions “easy to use” will ultimately lead to frustration and abandonment. While end-user composition is not a panacea for all of the troubleshooting ills that will befall users of increasingly complex networks of devices and services, it stands to reason that a system that allows users to express their compositions using constructs that resemble the system’s own constructs will give users a firmer basis upon which to later understand problems that arise.

2.5 Universal Remote Control

In terms of providing user interfaces for an integrated user experience, there is relevant work coming from both industry and research sources using the metaphor of a universal remote control.

2.5.1 Infrared-based Universal Remotes

The model promoted by industry is that of universal remote control—handheld devices that can be used to control a variety of different A/V components. Traditionally, these devices have been regarded as extremely cumbersome to configure and use, but current best-of-breed devices like the Logitech Harmony [18] have significantly eased the pain of the initial setup by allowing programming codes to be downloaded into the device directly from the Logitech website. They have also improved the user experience by allowing single button presses to send commands to multiple devices to set up the whole system to carry out an activity like "Watch a DVD," or "Listen to CD." The Harmony, like all other universal remotes, is not capable of making ad-hoc connections among networked services. It is also designed to only make device-device connections, and has no notion of interacting with content. As media becomes virtualized and is capable of being aggregated and served by large-scale content services, it will become increasingly important for users to be able to interact with content as a first-order entity. In contrast, Obje and OSCAR transparently support the intermixing of devices and content and the experience of content as a first-order entity.

2.5.2 Software-based Universal Remote Control of Networked Devices

Software-based universal remote control systems such as the Pebbles Personal Universal Controller [114], UNIFORM [113, 115], Huddle [116], the Trace Universal Remote Console specification [155], the Stanford iRoom's iCrafter [121], and the User Interface Markup Language [3] seek to improve upon infrared-based universal remotes like the Harmony by allowing devices to provide abstract descriptions of their capabilities and allowing client-specific UIs to be generated automatically based on those descriptions. Despite these advantages, these technologies are limited by the same factors that limit commercial industrial remotes: they cannot create ad-hoc connections and they do not address content-to-device interactions.

Roadie [89] provides a mixed-initiative user interface that allows users to specify high level goals such as "Watch a movie on DVD" and attempts to automatically formulate step-by-step plans that will enable the accomplishment of the goal. Where Roadie is unable to carry out a step by itself, it gives the user instructions about what needs to be done. In that the user interaction is based around the user expression of high-level goals that are mapped to system configurations by an internal planning algorithm, Roadie resembles the InterPlay

[100] and Task Computing [98] systems mentioned earlier. As these approaches trade end-user composition for automated composition, they provide users with less control and are less able to adapt to unconventional user needs.

2.5.3 Providing Devices with Web Presence

HP's Cooltown project [83] was concerned with extending web presence to devices and services situated in the physical world. While it was not the stated goal of Cooltown to enable "remote control" of devices or services, but rather to provide location-bound services and devices with points-of-presence on the World Wide Web, the effective user experience of using devices in Cooltown is very similar to the experience of universal remote control. The user's client device (in this case running a web browser) obtains access to a control channel to the device (in this case by obtaining its URL through an out-of-band mechanism such as barcode scanning or RFID reading) and thereby presents the user with a set of controls (in the form of a web page) for the device. The device is free to provide any form of control or content via its web interface that it wishes—no standard controls, service interfaces, or description languages are dictated by Cooltown. While this should make it relatively easy to create and deploy new services, as web technology for both servers and clients is extremely common and familiar to most software developers, it does not ensure any consistency of user experience across services and it does not support end-user composition of multiple services. Each service is a standalone island of functionality that cannot be easily combined with any other island's functionality.

2.5.4 Shortcomings of Universal Remote Control

The primary disadvantage of the universal remote control approach is that it only supports the one-to-one association of clients and devices for the purpose of control. If a user wants to create or control a connection between two or more devices, as is the case when for example media from one device is being streamed to and rendered on another device, the universal remote—whether software or hardware—will be unable to assist with that process. What is required instead is that the connection be made in advance through a separate, out-of-band mechanism (such as connecting the devices with dedicated A/V wiring), at which point the universal remote can activate or deactivate the connection by treating the connection as a control parameter of *one* of the devices. For example, if a DVD player is connected to a TV's "AUX" port, the universal remote will be able to effect the connection between the two by turning on the TV, turning on the DVD, and then switching the TV's input to "AUX." If the DVD player had not been previously connected to the TV in this fashion, the connection would not be possible using the universal remote. Even the software remote control systems described in this subsection have not addressed this concern, and remain unable to forge ad-hoc spontaneous connections among devices that do not have a previous association with one another.

2.6 End-User Composition of Networked Services

End-User Programming is a long-established field of research that looks at how to provide end users with the tools, skills, and motivation to create their own functionality. The huge number and wide variety of end-user programming systems (for a thorough survey, see [79]) precludes a thorough treatment of all of them here, so I will instead focus on the subset of end-user programming systems that have looked at providing users with the means to construct new functionality out of networked services.

These systems share two key characteristics: their emphasis tends to be compositional rather than behavioral and they must adapt dynamically to changing conditions in the environment. To say that they are *compositional* rather than *behavioral* in emphasis is to highlight that each of these systems is aimed at providing users with the ability to assemble blocks of predefined functionality, rather than create functionality from the ground up. In this sense, they resemble the domain-specific programming languages [99] (such as the Excel formula language [159]) advocated by Nardi in her seminal study of end-user programming [109] rather than general purpose programming languages such as Java [144] or even languages such as Logo [90] and Visual Basic [101] with explicit aims to be accessible to a wider range of end-users than traditional programming languages. To note that these systems must *adapt dynamically to changing conditions in the environment* is to say that they have the additional challenge of presenting to users different sets of functionality at different times, depending on the compositional components that are available for use. This is different from traditional end-user programming systems where the set of functionality remains fixed unless the system has been explicitly upgraded by the user.

In addition, all of these systems happen to share the characteristic that they are primarily targeted for domestic environments, which suggests the additional shared characteristic that they must be extremely easy to learn and use since it is presumed that domestic users are less motivated than professional users to learn and master complex software in the hopes of an eventual return on their investment of time and effort.

2.6.1 End-User Programming in the Digital Home

The AutoHAN Media Blocks [9] provide a tangible user interface to allow configuration and control of domestic appliances. Each block represents an operation, object, or programming construct. Users construct configurations by arranging the pre-assigned blocks to construct statements. The blocks are also capable of being used directly with certain devices, so for example a "play" block is able to be associated with the "play" command on any device that supports such a command, such as a CD player or VCR.

The Jigsaw Editor [70, 127] employs a jigsaw puzzle-based metaphor for snapping together functionality. This metaphor is built and studied in both a tangible and graphical form. Like AutoHAN's blocks, jigsaw puzzle pieces represent devices and data items, and users may create rules by connecting them into linear chains that are then mapped into system configurations that are evaluated and triggered when appropriate. The paper does not exhaustively describe the capabilities of the compositional system; indeed their system was developed in large part to engage end-users in participatory design activities to determine what capabilities are needed for a system that allows end-user programming for "smart home" behavior.

CAMP [150] takes a similar approach to AutoHAN and the Jigsaw Editor in that it allows users to compose functionality by selecting from a predefined set of operators and objects and arranging them into sequences in order to express rules and compositions. In the case of CAMP, a metaphor of magnetic poetry is used, and the operators and objects are English words and phrases rather than blocks or iconic representations. CAMP focuses on allowing users to create capture and access applications, and thus restricts the domain of interest somewhat. Within this domain, however, CAMP is able to express more sophisticated compositions, and is able to express them in less ambiguous terms than the other two systems.

iCAP [33] provides a graphical UI that allows users to define conditional (if-then) rules by dragging and dropping icons representing inputs and outputs onto the appropriate graphical region of the UI window. Unlike the previous systems, the user is in charge of defining and parameterizing the inputs and outputs, and in assigning them a graphical representation. This would appear to offer a high degree of flexibility in terms of defining arbitrary inputs and outputs, though the available options for creating these entities are not described in detail. Also, the restriction to define compositions as if-then statements restricts the types of compositions that can be created.

In each of these projects, the focus is largely on the user interface for specifying a composition, and less on other aspects of interacting with compositions such as storing and organizing compositions for later access, invoking previously created compositions, understanding and modifying existing compositions, or sharing compositions with others. One exception to this generalization is the AutoHAN system, which uses a language-independent script representation called Lingua Franca [59] to store programs created by the Media Blocks. These scripts can, in principle, be translated to and from a variety of programming languages including textual scripting languages, visual programming languages, and tangible languages such as the Blocks. Consequently, the scripts may be manipulated in different ways and possibly by different individuals.

Lingua Franca provides a mechanism for user-created compositions to live beyond their initial creation, and this capability foreshadows the most significant shortcoming of all work related to end-user composition or programming in the home, which is a failure to account for the entire lifecycle of user-authored functionality. In the next section we present our view of the end-user composition lifecycle and examine how this viewpoint affects our understanding of the requirements for systems that intend to support such activities.

2.6.2 Design and Prototyping Tools for Ubicomp Applications

Another body of work that shares an affinity with end-user composition is the research surrounding design and prototyping tools for ubicomp applications. Like end-user composition systems, such tools are focused on allowing non-developers to create applications quickly. These tools differ from end-user composition systems primarily in that their intended audience is designers who are typically producing or prototyping applications for someone other than themselves, whereas end-user composition systems are primarily focused on allowing users to create new functionality for self-benefit.

While not explicitly compositional, location-based experience design tools like Topiary [88], MOPS [15], and the Mobile Bristol toolset [69] allow designers to craft behaviors that are based on information about users' locations and other contextual factors. The information about the users' location can be obtained from a variety of sources, and to the extent that this information will ultimately be obtained from external sources such as sensors placed in the environment, the tools are facilitating the design of a composed application.

The "mash-up" is a form of web service composition that is rapidly gaining popularity [64]. In essence, a web mash-up is a new web service or site that incorporates data from two or more other web sites using the sites' web service API or by creating a programming interface to the web site through the use of "scraping" tools that parse a site's HTML output and allow programmatic access to the site's functionality. End-user- and designer-focused mash-up creation tools are beginning to emerge both within the research community [12, 62, 166] and outside of it [85, 102, 168], though usage experiences with these systems have not been reported extensively in the literature up to this point.

2.6.3 Shortcomings of Existing End-User Composition Systems

The systems presented in this final subsection of Chapter 2 are aimed at providing end-users with the tools to create functionality they need by composing services that may be distributed among various hosts in the local- and wide-area network. As such, they come the closest to providing the user experience that is required for emerging ubicomp environments—that is, a customizable, integrated user experience of a large and rich network of devices and services. However, these projects share two key shortcomings: first, with the exception of the work on web service mash-ups, they have not been linked to generalizable service frameworks that would allow them to be deployed in a realistic setting outside the laboratory. Rather, they have been developed and tested with constrained sets of custom-built services which were tailored specifically for the scenarios they were designed to support. Secondly, most of them have not been tested with users in a realistic setting with live, reactive services. For example, the user studies for the Jigsaw Editor, iCAP, and CAMP, while certainly informative, all used mockups that were not connected to the underlying infrastructure and therefore did not allow users to experience the effects of their compositions in anything but an abstract, imagined way. Other systems, such as Topiary, that have conducted user studies with realistic, simulated, or wizard-of-oz data have been aimed at application designers rather than end-users designing for their own consumption.

Some of the systems described under the heading "End-User Programming for the Digital Home" could perhaps be adapted for use with Obje devices and services. In particular, the Jigsaw Editor, CAMP, and iCAP are sufficiently general in their scope that the ability to make data-centric connections along similar lines to the connections created and managed by OSCAR could be implemented without significant difficulty. However, none of these systems as they have been designed provide facilities for ad hoc discovery or control of components, distinguishing between one-off, temporary connections and persistent, reusable compositions, or ongoing monitoring and control of active connections. AutoHAN is even more limited because of its exclusive focus on tangible input—adaptive control and monitoring is essentially impossible without significant redesign. Thus, even if these UIs could be adapted for use with Obje, they would have to be significantly expanded and redesigned to support all of the end-user composition and control features that are supported by OSCAR.

The systems described in "Design and Prototying Tools for Ubicomp Applications" are, for the most part, similarly amenable to being adapted for use with Obje services. However, these systems are not designed to be used by the ultimate end-users of the functionality they allow to be created, and so have many aspects (e.g., support for authoring of complex, abstract behaviors and support for Wizard-of-Oz user testing) that are suitable for designers but much less so for end-users.

2.7 Summary

I have described a wide range of work that has addressed various aspects of providing a seamless, adaptable user experience of interacting with multiple devices and services, and I have shown that, thus far, there has been a shortage of work that adequately pulls together the required pieces to provide such an experience. In the next chapter I will present the Obje Framework, a robust interoperability platform that is designed to support interoperability in the face of open evolution of various device and service types and communication standards, and is designed to enable end-user composition of devices and services. Following the presentation of Obje, I will present two case studies of Obje applications, including their design, implementation, and use, and will show how those experiences shed light on the design of integrated user experiences of ubicomp environments.

3 The Obje Framework

1.1 Introduction

For users to have an integrated experience of the range of interconnected devices and services that comprise their physical/digital environment, it is necessary that the components of that environment work together reliably and consistently, with minimum effort on the part of the user. The goal of a system that supports end-user composition is to enable users to remain in control of **what** is to be connected and **when**, and to reduce as much as possible the necessity of users' involvement in negotiating **how** they connect. Thus it is a key role of infrastructure to facilitate the interoperability of devices and services in ubiquitous computing environments. In addition, infrastructure must provide basic facilities, such as allowing users to discover, identify, comprehend, interact with, control, and extract information from the services they encounter.

In this chapter, I describe the Obje Framework[3], which is an infrastructure for ubiquitous services that provides all of the facilities described above. Some of

[3] This chapter describes joint work carried out principally with Keith Edwards, Trevor Smith, and Jana Sedivy. Portions of the work described in this chapter have been previously described in [42] and [112].

these facilities are not provided by Obje in a manner that is particularly unique or interesting. For example, the mechanism for discovering services in Obje is based on the zeroconf standard which is the same as the mechanism used by Apple's Bonjour [4] and isomorphic to the mechanisms used by UPnP's SSDP [50] and Jini's Discovery protocol [143]. However, there are two key areas where Obje's approach differs from all other approaches that have been taken in this domain. One area is Obje's approach to interoperability, which allows runtime negotiation of many details of interoperation through the leveraging of "meta-interfaces" combined with distribution of mobile code. The other area is Obje's ability to facilitate the delivery of user interfaces to various client devices in a way that allows services to "push" control UIs to clients at particular stages of their interaction with the user and with other services, and that allows clients to present integrated UIs for the simultaneous control of multiple devices without requiring specific advance programming or configuration of those clients.

3.1 The User Experience of Interoperability

Many scenarios of mobile, pervasive, and ubiquitous computing envision a world rich in interconnectable devices and services, all working together to help us in our everyday lives. Indeed, we are already seeing an explosion in new, networked devices and services being deployed in workplaces, the home, and in public spaces. And yet, in reality it is difficult to achieve the sorts of seamless interoperation among them that is envisioned by these scenarios. How much more difficult will interoperation become when our world is populated not only with more devices and services, but also with more types of devices and services?

These visions of ubiquitous computing raise serious architectural questions: how will these devices and services be able to interact with one another? Must we standardize on a common, universal set of ubicomp protocols that all parties must agree upon? Will only the devices and services that know about one another, and are explicitly written to use one another, be able to interoperate, while others remain isolated? How will such networks be able to evolve to accommodate new devices that may not have even existed at the time current devices were created?

These architectural questions in turn pose pressing issues for the user experience of ubiquitous computing. Architectures that impose such constraints, that prevent us from easily adapting and using the technology around us for the purpose at hand, cannot support the sort of "calm technology" envisioned by Weiser and Brown [161]. Instead, they are likely to bring the frustrations of software incompatibility, driver updates, communication problems, and version mismatches—limited islands of interoperability, with few bridges among them.

As will become clear shortly, these weaknesses are inherent in current architectural approaches to interoperability. Overcoming these impediments requires new approaches in which devices and services can interact with peers without having to know about them ahead of time. Such an approach would allow devices to work with one another without requiring replacement, upgrade, or patching, allowing networks to evolve to accommodate entirely new types of devices easily. In other words, such an approach would remove the system-imposed constraints on interoperability that prevent us from freely combining and using the technology around us. If, on the other hand, the software in our devices must be updated to work with every possible new type of thing they may encounter, then we will be locked into a world of limited interoperability and the requirement for lockstep upgrades of the devices on our networks.

3.1.1 Patterns of Communication

Fundamentally, communication between any two systems depends on prior agreement about the interfaces supported by those systems. These take the form of the protocols, operations, data types, and semantics that, ultimately, two systems must be coded against in order to work with each other. For example, to interact with a service, a client must be explicitly written to “understand” the service’s interface—what operations are available, how to invoke them, and what the semantics of these operations are. In the case of the web, for instance, there is agreement on the form of communication (HTTP), the format of data exchanged (HTML, mainly), and the semantics of that data (most clients will issue a GET request when they encounter an IMG tag, for instance). In the case of Universal Plug’n’Play (UPnP)[74], these agreements include not just the generalities of

UPnP itself (such as the SOAP protocol, and device descriptions expressed in XML), but also the device-specific profile used by a particular UPnP device, which defines the set of allowable operations for that device type.

Current approaches to interoperability take what might be termed an ontological approach—a standard is created that defines how a device's functionality may be exposed to and accessed by peers on the network. Knowledge of this standard—in the form of software that implements its required protocols, data formats, and semantics—is then built into peer devices on the network. New types of devices, which do not sufficiently resemble the existing ontology to the point that they can be retrofitted into it, necessitate the creation of yet more interface definitions.

This same approach is taken by virtually all networked communication systems today. UPnP defines device interfaces, termed *profiles*, for a range of device types, including scanners, printers, media devices, HVAC systems, and so forth. Bluetooth likewise defines similar profiles for headsets, telephones, hands-free car systems, and so forth. Virtually without exception, new versions of these standards define new device interfaces (or revise existing device interfaces) that will be unknown to existing devices on the network.

Arrangements such as these gain interoperability at the expense of evolution. Because the agreements necessary for communication must be built in at development time to all communicating parties, the network cannot easily take advantage of entirely new types of devices. New types of devices must either be retrofitted into the existing interfaces in order to maintain backwards compatibility (at the cost of losing access to whatever specialized functionality the new device type provides), or new interfaces must be created that describe the new device type (at the cost of having to update all existing devices with the programming to communicate with the interface used by the new device type—or, more likely, by simply replacing the existing devices with newer ones).

Such models of communication seem fundamentally at odds with a vision of ubiquitous computing predicated on device abundance, in which new types of technology are easily deployed, integrated into the environment, and appropriated by users.

3.1.2 The Recombinant Computing approach

If communication requires such up-front agreement, then the central question posed by this research is: is there an approach to creating such necessary agreements that can better accommodate current devices and services while supporting evolution to accommodate future devices and services? Further, can this be done in a way that supports rich and useful interactions?

The conceptual foundation of the Obje Framework is an approach we call *recombinant computing*. The key insight behind recombinant computing is that robust, evolution-resistant interoperability can be obtained by shifting a portion of the knowledge necessary for communication from *development-time* to *run-time*. This shift is made possible by dictating an agreement on a minimal set of development-time interfaces that are then used to allow devices to negotiate further necessary agreements—along with the behavior needed to implement these—at runtime. The term “recombinant computing” is meant to evoke a sense that devices and services can be arbitrarily combined with each other, and used by each other, without any prior planning or coding beyond this minimal set of development-time agreements.

To achieve interoperability in this model, three criteria must be met for the base-level agreements that are built into devices at development time. The first is that the interfaces that devices support must be fixed, since this is what guarantees future compatibility. If the interfaces necessary for communication are allowed to evolve, then existing devices may be unable to speak the later versions of the interfaces supported by newer devices. Second, the interfaces must be minimal. Otherwise they are unlikely to be adopted and implemented fully by developers, imposing new barriers to interoperability. Finally, such a fixed and minimal set of interfaces will necessarily be generic, since any small fixed set of interfaces cannot capture all possible device-specific notions, for all unforeseen types of devices.

The problem, however, is that relying only on a small set of static, generic interfaces is incredibly limiting, and supports only simplistic interactions among devices. For example, although one could achieve interoperability by dictating that all parties on the network, no matter what their function or semantics, exclusively use plain text over FTP as their communication mechanism, such an arrangement clearly limits the range of possible applications.

Therefore, our system couples these static development-time interfaces with the runtime exchange of new behaviors over the network. In essence, the fixed agreements become *meta-interfaces* that, rather than dictating how a service and a client interact with each other, dictate the ways in which the service and client can acquire new behavior that allows them to interact with each other. This new behavior takes the form of mobile code that is provided by devices on the network to their peers at the time of interaction.

The approach is reminiscent of Kiczales, et al.'s metaobject protocols [82], in which a set of fixed interfaces can be used to provide runtime extensibility (in Kiczales et al.'s case, of programming languages). Here, however, new behaviors are provided over a network connection in the form of mobile code, rather than simply across a procedure call boundary.

Using this model, devices only build in the most generic agreements (the interfaces for acquiring and invoking mobile code), and defer other agreements necessary for interoperation until runtime. When new devices appear on the network, they provide certain behaviors to existing peers to bring them into compatibility, essentially "teaching" their peers how to interact with them.

The recombinant computing approach is fundamentally different than the standard "ontological" approach based on detailed a priori agreements about all aspects of a device's syntax and semantics. Approaches based on extensive and changing ontologies (or "profiles," or "service descriptions") place the burden of work inappropriately: when a new type of device appears, every single other device on the network must be updated to work with it. In an approach based on runtime agreements, in which functionality needed for interoperation with a new device is provided by the device itself, the burden of work is borne only by that new device.

3.1.3 The Role of User-Supplied Semantics

Ad hoc interoperability poses important implications for the user experience. Under current ontological approaches, if a device can interoperate with a peer at all, one can reasonably expect that the device understands the semantics of that peer—what it does, when to use it, and so forth. A Bluetooth phone, for example, "knows" what a Bluetooth headset is capable of, and when to use it (when a phone call happens, connect to the headset and stream audio between it and the phone) because this understanding of how to seamlessly mesh the semantics of the two devices has been built into them by their developers, allowing rich and semantically-informed interactions between them.

Such is not the case in a world of ad hoc interoperability, however. If applications can only interact with the things with which they are expressly written to interact, then new types of devices will be inaccessible to them. Instead, the common case will be that applications will encounter new types of devices, about which they have no special semantic knowledge: they will be able to communicate with such devices, yet without necessarily knowing what the new device does.

For example, under a model of ad hoc interoperability, a phone may detect that there's a device present that it can communicate with, and perhaps send audio to, but may not be expected to know whether that device is a headset, a speaker in a public place, a storage device, or a service for transcribing speech to written text. Nor arguably should the phone have to have such knowledge in order to communicate with the new device since, if interoperability is our goal, we do not want to require that devices only be able to interact with things they already know about.

If devices do not contain the specialized programming necessary to allow them to work in semantically-informed ways with specific types of peers, then it will sometimes, perhaps even often, fall upon the user to provide the semantic interpretation and arbitration missing in the devices' programming. In the example above, it must be the user who is tasked with understanding what a particular device does, when it makes sense to use it, and so forth; the role of the infrastructure, under this approach, is simply to allow the interaction to take place, should the user decide to do it. This is an example of what Fox and Kindberg have termed "crossing the semantic Rubicon"—requiring users to take on some of the burden of semantic interpretation that once was the domain of applications [84].

The implication that users must be "in the loop" in determining when and how to use newly encountered devices is inherent in any model of ad hoc interoperation, and it is central to the design of Obje. Users will typically have the role of providing the semantic interpretation that may be missing in the programming of the application. This requirement has ramifications on the sorts of user experiences that we can create, which in turn has implications for the design of the infrastructure that must allow users to easily understand and use the resources around them.

3.2 The Obje Framework

This section describes the basic design of Obje. We first examine the low-level capabilities that we expect to be built into devices and applications that implement the Obje software stack; then we describe how these low-level capabilities are wrapped in a programming model that supports the creation of dynamically extensible devices and applications.

The most distinguishing feature of our middleware platform is that it allows runtime extensibility of devices and applications, allowing new devices that enter the network to provide code to peers to allow them to interoperate with the new device.

This function is exposed through a bootstrap protocol, layered on top of TCP/IP that provides a number of operations designed to support runtime extensibility. Most importantly, the protocol allows a new device on the network to provide a peer with:

- An implementation of a new network protocol needed to communicate with the device
- An implementation of one or more type handling modules, to render or process media or other data received from the new device
- An implementation of new user interface controls, which can be used to control the device remotely
- A transparent bridging mechanism that can allow a peer to acquire new discovery protocols, or the ability to interact with devices that may not exist on the IP network, or that may not support the Obje software platform

We call this protocol a bootstrap protocol because it is used for the initial negotiation and transfer of new capabilities necessary for compatible communication. Once this transfer has been completed, two devices communicate with each other directly using these new capabilities; the bootstrap protocol is not used for further communication.

These implementations of new capabilities are in the form of mobile code—self-contained executable content delivered over the network to the peer device. Our middleware platform allows for a variety of code formats, including platform-independent code (e.g., Java bytecodes) as well as highly-tuned, platform-specific code (which of course would only be executable on a compatible target device).

To participate natively in the Obje platform, devices must carry an implementation of the bootstrap protocol, may have one or more versions of mobile code intended for use by peers (these would typically be carried in some form of stable storage, such as firmware, flash, or on a disk), and may optionally have the ability to execute code received over the network. As we explain in later sections, our architecture also provides for non-Obje devices to participate in the platform indirectly, through proxies provided by a host computer or other device.

1.1.1 Bootstrap Protocol and Code Formats

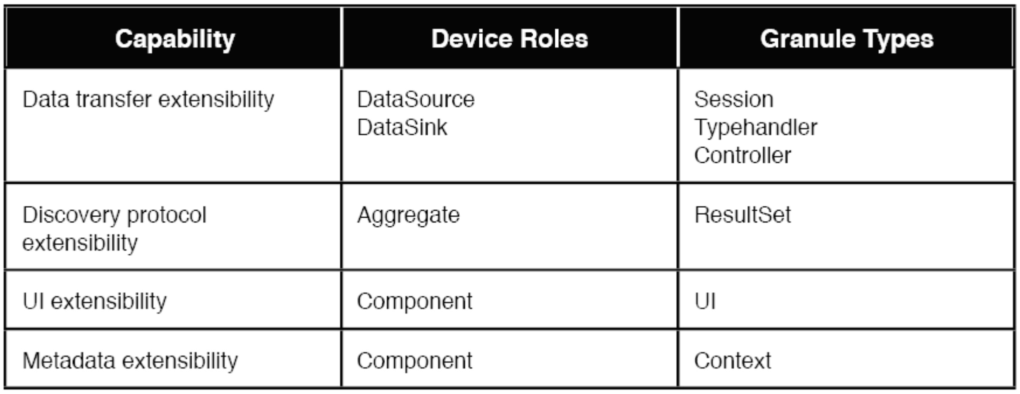
The Obje bootstrap protocol is defined as a profile on the Blocks Extensible Exchange Protocol (BEEP) [132], a generic application protocol framework for bidirectional, connection-oriented communication. BEEP provides facilities for Transport Layer Security (TLS), which provides message integrity and privacy as well as authentication of peers on the network.

Devices advertise their presence over the local link via Zeroconf [72], a widely-supported discovery mechanism based on multicast DNS (mDNS) [22] and DNS Service Discovery (DNS-SD) [21]. These advertisements take the form of Uniform Resource Identifiers (URIs) that indicate the address of the device. Once a URI for a given device has been discovered, an Obje peer may communicate with it using the bootstrap protocol. This initial communication comprises a FetchRequest message to the peer, which responds with its ComponentDescriptor. ComponentDescriptors are short XML documents that provide descriptive information about a device (name, icons, and so forth) as well as information about which roles the device may play (source or recipient of data, and so forth, as described below), and any mobile code that may be provided by the device.

These stand-alone bundles of mobile code are called *granules*, and they are represented in the protocol by elements called GranuleDescriptors. Each GranuleDescriptor indicates a location from which the mobile code may be loaded (typically, from the device itself), as well as parameters used to initialize loaded code granules, a unique version identifier that may be used by clients to cache code granules, and a specification of the platform requirements of the code granules.

Devices that can send or receive data declare in their ComponentDescriptors any content types that they may be able to process “natively” — meaning, without the need to acquire any code granules from a peer in order to interpret the received data. These declarations are in the standard MIME format [13]. This mechanism allows for the creation of devices and services that can participate natively in the Obje platform, but do not require the ability to download and execute mobile code, albeit at the cost of losing runtime extensibility. For example, a small viewing device may declare that it can accept JPEG and PNG image data only, and refuse to accept (or be unable to process) granules that could extend its type handling behavior to other image formats.

Depending on the roles a device plays with respect to its interoperation with other devices, it may provide a number of types of granules for specialized uses. For example, devices that play the role of originating data (called DataSources) may be able to transmit specialized granules that provide new protocol implementations or new type handling behavior (such as new CODECs) as described in the section Data Transfer, below. Other sorts of devices may provide custom UI implementations or custom discovery protocols other than Zeroconf (see the sections User Control and Metadata and Aggregation, respectively). There are a fixed number of device roles defined by Obje, and thus a fixed number of granule types. Devices that play a given role are written to provide or accept the granule types defined by that role.



Capability	Device Roles	Granule Types
Data transfer extensibility	DataSource DataSink	Session Typehandler Controller
Discovery protocol extensibility	Aggregate	ResultSet
UI extensibility	Component	UI
Metadata extensibility	Component	Context

Table 3-1: The four primary modes of extensibility defined by Obje are described as four "roles" in which devices can be used; devices that play one or more of these roles can provide or use a fixed set of granule types. Devices may participate in multiple roles.

Table 3-1 shows an overview of the different device roles and the corresponding granule types used by those roles. Device that can participate in data transfer implement one or both of the DataSource or DataSink interfaces; such devices support extensibility over support various aspects of data transfer, using Session, Typehandler, and Controller granules. Devices that can provide access to other devices—for example by encapsulating a new discovery protocol—support the Aggregate role, which provides extensibility in how peers acquire access to other devices on the network through ResultSet granules. Devices that provide access to custom user interfaces to control them, as well as to descriptive metadata, participate in the Component role, which is considered a base-level role that all devices should support; these devices use UI and Context granules to support extensibility along these dimensions.

Once a granule, such as a new protocol implementation, is transferred to a peer, it can be executed directly by that peer. Thus, after the initial bootstrap phase to exchange any code necessary for compatibility, two Obje peers can communicate directly with one another, using the protocol and data types provided by the source device. Thus, further communication does not involve the bootstrap protocol itself.

Effectively, this mechanism allows peers to be built against a static protocol specification (the bootstrap protocol), which is then used to exchange new capabilities necessary for compatibility (in the form of granules), as new peers enter the network. It is this approach that allows devices to be coded against a fixed protocol, and fixed set of granule types, and yet be extensible to support new devices encountered "in the wild."

3.2.1 The Obje Programming Model

We have developed a programming model that wraps the low-level boostrap protocol, along with other aspects of our infrastructure, including remote code loading and discovery extensibility. This programming model not only allows easier creation of applications, but also maps the capabilities of the platform into a polymorphic, object-oriented framework: devices on the network appear to applications as objects in their local address spaces; the roles those devices can play are mapped on to the fixed set of meta-interfaces in our conceptual model;

the methods in those interfaces use and return objects that themselves implement well-known interfaces, and which are implemented by the granules returned from devices on the network. Thus, to applications, the loading of code granules is transparent, appearing simply as new, polymorphic implementations of already-known interfaces.

This approach of transferring necessary implementations of known interfaces across the wire, rather than through standard single address-space method calls, is similar to the model used by Java's Remote Method Invocation (RMI) framework [164]. However, our implementation differs from RMI along a number of key dimensions. First, it is not specifically tied to the Java programming language, and can support mobile code in a number of formats. Second, it neither provides nor requires the distributed garbage collection facilities of RMI; code transferred to an application is used to create a new instance of an object in the application's address space, rather than a reference to a remote object that must be factored in to a reachability analysis for distributed garbage collection. Third, it does not use serialization (which can often be fragile, especially in the face of object versioning or the need for multi-platform support) to transfer instance data across the wire; only implementations are moved.

Devices are represented in Obje by components, which are simply objects that reside in the address space of the client applications that use them. Components can be thought of as proxies for accessing a device such as a projector, printer, or

PDA. This programming model is implemented by a small messaging kernel, against which applications are linked. The messaging kernel implements the Obje bootstrap protocol, and is responsible for creating new component proxy object representations within the client application's address space; this representation is created from information contained in the device's ComponentDescriptor. The kernel, upon receipt of the ComponentDescriptor from a device, generates a proxy object that represents the new device, and notifies the application that it is available via a simple event interface.

The component proxy objects that are generated by the kernel implement one or more programmatic interfaces that allow applications to access information about the remote device, as well as to acquire mobile code from it. Thus, while applications operate on these component objects using normal local method calls, these calls are translated into wire messages in the bootstrap protocol by the messaging kernel, and are sent to the backend service or device. For example, invoking one of the data transfer-related interfaces on a component (as described below) causes a request for the necessary granule to be encapsulated into the bootstrap protocol and sent to the remote device, which then returns the code to the client.

Figure 3-1 illustrates the process. Here, the messaging kernel in the application first discovers the device, and then acquires its ComponentDescriptor, which the kernel uses to create a new component proxy object. The application can interact with this new component object to query its name and other descriptive information, as well as to obtain mobile code from it that can be used to extend the application's behavior in certain prescribed ways.

When mobile code granules are transferred to a client device or application, the messaging kernel expose one or more of the granule-provided objects that implement a set of known interfaces that define the ways in which client applications can interact with new implementations to specialize the client's behavior. The objects provided by the messaging kernel are simply the instantiation of a granule loaded across the network (in the case of a Java-based granule), or a wrapper object (in the case of a native code-based granule). Thus, applications written against the high-level programming model never see granules directly, but rather normal Java objects that implement well-known interfaces, but whose implementations come from granules delivered over the wire.

Figure 3-2 illustrates the process of acquiring and using granules from the application's perspective. Here, application code interacts with a component, which proxies for the device shown on the right. Local method calls on the component cause the messaging kernel to request necessary granules from the backend device, which are then returned to the application; both the request and the response are transmitted over the bootstrap protocol. Once the granules are returned, the messaging kernel wraps them in objects that implement interfaces assumed to be well-known to the client, where they can be operated on using local method calls. In the case shown here, granules provide a custom protocol implementation, custom data type handling code, and a custom user interface for interacting with the device; these granules represent polymorphic implementations of the interfaces known to the client, transferred over the wire from the originating device. Once the new protocol granule has been loaded and wrapped as an object, the application communicates with the device using the protocol implemented by that granule, rather than the bootstrap protocol.

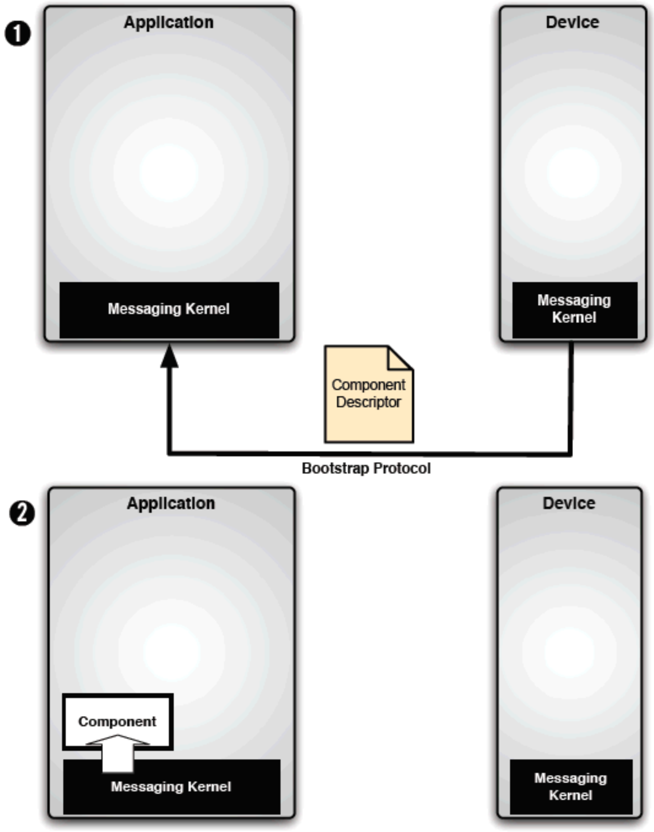


Figure 3-1: The application on the left discovers the device on the right. A Component Descriptor, passed via the bootstrap protocol, encapsulates information describing the device (1). The Component Descriptor is used by the messaging kernel in the application to create a component representation as a proxy for the device (2).

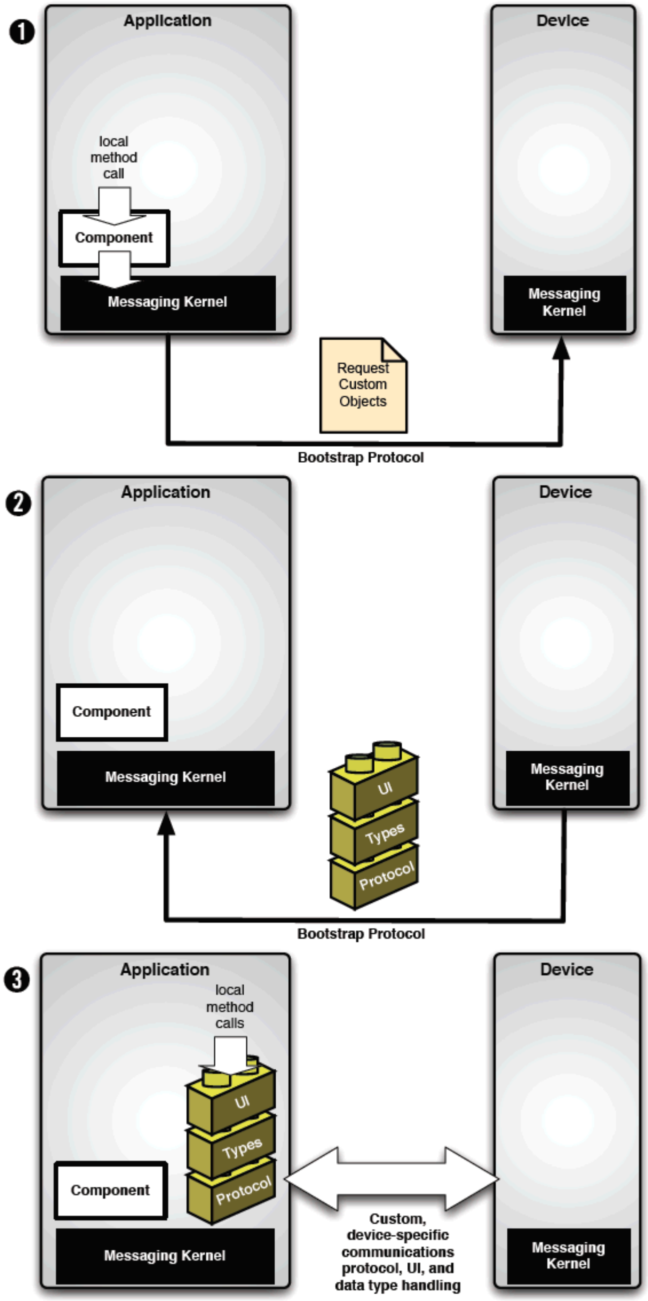


Figure 3-2: The application interacts with the component to retrieve granules from the device, which allow the application to specialize its behavior for protocol, data type, and UI handling. Here, the application invokes a local method call on the the component (1) that causes it to request a granule from the backend device. That device returns granules that provide new protocol implementation, data type handling behavior, and UI specific to controlling communication (2). These are loaded into the application and returned as custom objects via the local method call on the component, where they can be used by the application.

In our current implementation, the messaging kernel is implemented in Java, and produces component proxy objects that implement a small, fixed set of Java interfaces corresponding to the device roles described above. Thus, our high-level programming model most easily supports applications written in Java, although as noted we do support transfer and loading of native mobile code (described later in the section Experiences with the Obje Framework). Because the core protocols and wire formats of Obje are language-independent, devices and applications can be written against the bootstrap protocol directly, in languages other than Java.

The next sections describe the patterns used by devices in specific roles to support extensibility of data transfer, discovery, user control, and metadata.

3.2.2 Obje Service Roles and Communication Patterns

This section describes the patterns used by devices in specific roles to support extensibility of data transfer, discovery, user control, and metadata. These patterns and roles are the embodiment of the meta-interfaces described above in the section “The Recombinant Computing Approach.”

3.2.2.1 Data Transfer

The most important (and most complex) Obje mechanisms are those that support extensible data transfer between components, such as a PDA sending data to a printer or a video camera sending a video stream to a fileserver. These mechanisms have been successfully used in a wide range of devices and client applications (see the section The Experiences with the Obje Framework for details), and provide runtime extensibility along three important dimensions: extensibility to new protocols, extensibility to new data types, and extensibility to new user interfaces for controlling a data transfer. The three sections below discuss each of these in turn.

3.2.2.1.1 Protocol Extensibility

Obje devices can play two roles in a data transfer: data sources and data sinks. These roles dictate how the devices will exchange mobile code during a data transfer: data sources provide new mobile code-based protocol implementations, which are then used by data sinks to retrieve data from the source using the new protocol. In the Obje terminology, this new protocol implementation is carried in a type of granule called a session.

Many connections between devices are initiated as a result of some user action at a client, the basic pattern of use supports transfers started by a third party such as a browser or other application. In this pattern, a client application requests a session granule from a source. The client will then pass this session to a data sink component to start the transfer. From this point, the source and sink exchange data directly, without it passing through the client. Figure 3-3 illustrates how a session is passed from source to sink by way of a client application that initiates the connection.

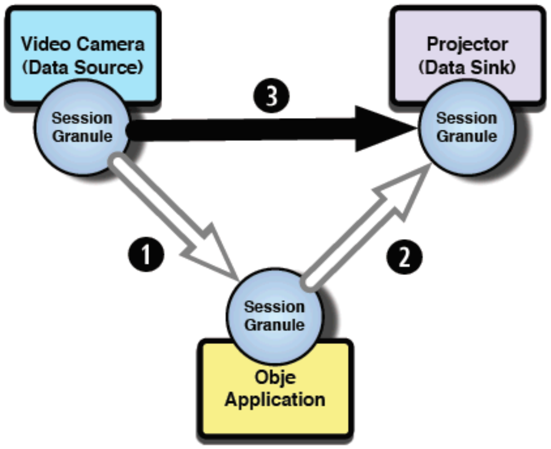


Figure 3-3: Data transfer initiated by a client application

Note that the act of requesting a session from a source, as well as passing it to a sink, will involve a number of messages in the bootstrap protocol. Specifically, this sequence of operations will cause a request for the session granule to be sent to the remote source device, the mobile code for the session granule being returned over the network from the remote source to the client, and then passed from the client to the remote sink. The Obje messaging kernel performs these requests as the client invokes operations on the component objects in its address space, hiding the details of them from the client.

In terms of our programming model, data source components provide a method that allow clients to list the formats of the data they provide (in the form of MIME types [13]), as well as a method to return a new session granule from the device. Once a client has retrieved the session granule from the source device, it can pass it to any data sink device, through a method defined on sink components. The session granule is then marshaled, and passed over the network to the receiving device, which unmarshals it and invokes the code within it to read data from the original source.

Since the source device provides the session granule, it effectively has control over both endpoints of the communication, allowing it to use whatever protocol is appropriate for the type of data being transferred. This process is transparent to the receiver of the data. In addition to providing protocol implementations, session granules also support a number of operations that allow clients to control the transfer of data between devices. Specifically, clients can terminate a session (stopping the flow of data), and can also subscribe to receive notifications about changes in the state of the transfer (that it has failed, for instance). In essence, the session acts as a capability, allowing any party that holds it to change its state, or be informed of changes in its state. Distribution of state updates happens in a semi-centralized manner: updates cause a message to be sent to the source that created the session, which sends the update to other holders of the session. In addition, each party's copy of the distributed session object monitors the connection with each of the other parties via periodic Heartbeat messages, and is thus able to detect quickly when one of the other parties has failed or crashed. Thus each party is able to recover from any other party's failure and restore its own state appropriately. As described below, this same mechanism is also used to asynchronously distribute user interface granules to devices involved in the transfer.

3.2.2.1.2 Data Type Extensibility

The features of the data transfer pattern outlined above—independence from specific protocols, the ability for third parties to initiate connections, and the ability to receive notifications about changes in the state of a transfer—are necessary but not sufficient for providing the flexible and seamless version of recombination that we envision. Namely, it allows easy, protocol-independent interconnections among components, but only insofar as those components understand the same data types.

Just as we believe that future devices will bring with them new protocols, an ability to work with new data formats and media types is also required. If components must be prebuilt to understand each other's data types in order to work together, we drastically limit their ability to interoperate in a truly ad hoc fashion, and to evolve to support arbitrary new devices. We need to move away from the requirement that devices must be replaced or manually updated each time a new data format appears on the network.

There are a number of approaches one might take to overcome this conundrum of extensibly handling new data types. One approach (reminiscent of Ninja's paths [52, 97]) would be to allow filter services (such as [120]) to exist on the network that accept data in one format and translate it to another. We intentionally avoided this approach, however, primarily because of its implications on the user experience of applications. If the data flow paths are created automatically, it may result in increased system complexity that is hidden from the user until such time as an error or breakdown occurs. At this point the user will likely have difficulty understanding the actions the system has already taken on his or her behalf and will be faced with a daunting recovery task. If, on the other hand, the user model requires users to explicitly connect a chain of format conversion filters, the user interface will reflect considerable complexity and in many cases ask the user to make decisions about data routing that they may not be able to fully comprehend.

Instead, the approach taken by Obje is to use mobile code to allow for extensibility to handle arbitrary data types, without the need for excessive user involvement, and especially without the need for a "wiring diagram" model of connection. Obje allows devices to broaden their statements of compatibility beyond simple MIME types. Specifically, devices declare the programmatic interfaces, or "representation class" by which they can provide or receive data in addition to the format of the data itself. For example, a projector—a device that by its semantics is designed to display things—might claim that it can understand not only JPEG data (expressed as the MIME type `image/jpeg;` along with the default representation class `java.io.InputStream`), but also that it understands the semantics of other things that are displayable (expressed as `application/x-obje-typehandler-granule;representationClass=”com.parc.obje.datatransfer.Viewer”`). In other words, a component might claim that it understands some abstract representation of a set of operations that it can perform on data (e.g., allocate a rectangle of its screen and invoke the method `Viewer.getPanel()` to allow a bundle of hosted mobile code to render arbitrary visual data), without having to understand the data itself.

By declaring that it understands a particular interface, a sink indicates that it is written to understand and use objects that implement that interface. Likewise, a source that declares that it can provide a particular interface means that it can transfer a specific implementation of that interface.

We call these mobile code-based implementations typehandlers, because they wrap a previously unknown data type in an object of a type known to a receiver. In essence, this extension of the type system allows sources and sinks to negotiate richer interfaces to the data they exchange and allows data type-specific implementations of these richer interfaces to be acquired at runtime. If both the source and sink agree on an interface they understand, a mobile code-based implementation of this interface will be transferred from the source to sink as a granule, and invoked by the sink to handle the data.

The set of interfaces that these typehandlers may implement is open-ended and extensible. This is the same as with MIME types: there is an extensible set of them, new ones will come over time, and they must be known to the involved parties for communication to occur, but no one else need understand them.

Figure 3-4 shows an example, displaying Powerpoint format data on a projector that is not explicitly written to use such data. In this case, the projector is written to understand a number of raw data types (GIF and JPEG, in our current implementation), and is also written to understand objects that implement an interface called Viewer. Any party that can provide a Viewer wrapper around its data can thus connect to the projector. Here, the projector downloads and uses a specific typehandler implementation of this interface that renders Powerpoint, a format previously unknown to the projector.

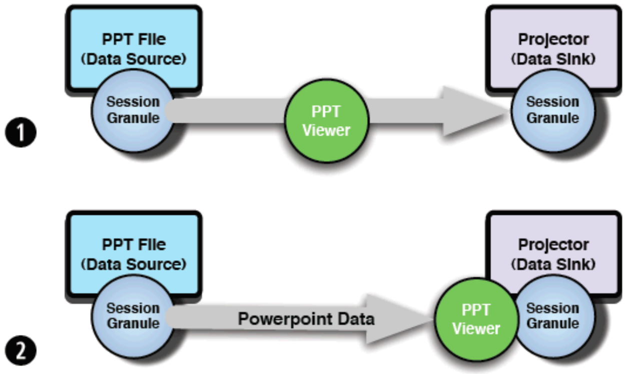


Figure 3-4: A projector device uses a typehandler to process data in an unknown format

The use of typehandler granules does not solve all problems with type compatibility— in the example above, even though the projector doesn't have to understand Powerpoint directly, it must still be written to understand the Viewer interface. The typehandler approach does, however, provide a number of concrete benefits over simply requiring agreement on simple data types. Most importantly, typehandlers provide a means for dynamic extension of the set of types that can be used between two devices. As long as a sink is written to accept a typehandler interface, components that were previously incompatible with it can be made compatible through the addition of a typehandler that meets that interface. Such easy, dynamic extension of type compatibility is not possible when only static types are allowed, without rewriting either source, sink, or both.

Such dynamic extensibility is especially important when a source provides an unusual format. For example, one of our current components provides a live video stream of a computer's display, using VNC [125]. We neither require nor expect all receivers to be able to parse and process VNC data, so the source provides a typehandler that implements a Viewer interface. Through this interface, any sink that is written to understand the semantics of Viewers can accept and display a live VNC stream.

Together the dimensions of runtime extensibility provided by the Obje data transfer mechanisms—extensibility to new protocols, new data type handling behavior, and new UIs (described below)—allow applications and devices to richly interact with each other, while requiring only minimal up front agreements about each others' interfaces.

Note that Obje's commitment to reducing the number and degree of a prior agreements required in order to work together means that Obje stops short of guaranteeing aspects of data transfer interoperation beyond a the basic ability to share and process data. For example, Obje does not make quality of service (QoS) guarantees, nor does it make any guarantees about the ability to, for example, synchronize multiple streams that a user may consider part of the same "logical" transfer. Since DataSources provide their own transfer endpoints, QoS guarantees can be made and enforced by the sources themselves, without requiring buy-in from the sink (though, admittedly, the source may in some cases require information about the sink's capabilities and performance that would ultimately require additional agreements [17]). The case of inter-stream synchronization requires agreements among the cooperating services, for example agreement on a globally shared clock [45] or designation of a master stream [122]. Such extensions could be implemented within Obje as a set of agreements that are made among typehandlers, but that remain opaque to the Obje bootstrap layer. For example, a SynchronizedViewer interface could be implemented that allows the typehandlers that implement it to participate in synchronized media transport sessions with other coordinated SynchronizedViewers, though again such a solution may require all cooperating sinks and services to provide additional information about their capabilities that is not currently made available through the Obje interfaces.

3.2.2.2 Aggregation

The second major group of mechanisms in Obje supports aggregation. Aggregates are devices that appear as logical collections of other Obje components. In Obje this pattern is used in a wide range of situations—to access filesystems, which appear as collections of components representing files and folders; to support devices that encapsulate new discovery protocols; and also to support devices that provide access to both devices on non-IP networks, and legacy (non-Obje) devices. Any situation in which a device provides access to other devices uses this interface.

From the perspective of our programming model, applications initiate an interaction with an aggregate component by performing a query() method call, defined by the Aggregate component interface. Applications pass a parameter that allows them to match devices based on their type, or metadata associated with them (see the section User Control and Metadata for details on device metadata), allowing applications to only be notified of a subset of available devices that pass some filter. The query operation returns a ResultSet to the requesting application. ResultSets are granules that present a simple dictionary view (component IDs as keys, and components as values) of the devices that match the query. Once an application has a ResultSet, it can iterate through the "contained" devices, and can solicit notifications about changes in the set of devices that the original query matches, allowing push-based notification of new devices. Thus, the semantics of

ResultSets are that they are “live,” and may be continually updated as matching devices come and go.

In our high-level programming model, initial Zeroconf-based discovery is presented to applications as a “root” Aggregate component that contains all Zeroconf devices on the local link, and supports the same query and notification mechanisms described here.

Because ResultSets are implemented as mobile code-based granules, this simple pattern can support a great deal of extensibility and power. For example, this mechanism allows applications to take advantage of arbitrary new discovery protocols, and to interact with devices on physical networks that the applications themselves do not have direct access to. The sections below describe how these scenarios work in Obje.

3.2.2.2.1 Discovery Extensibility

While Obje devices support Zeroconf as their standard discovery mechanism, such a “one size fits all” approach is untenable in a world of rich variety of devices and networked environments [38]. As a simple example, Zeroconf does not provide the ability to easily discover devices outside the local link; in such cases, it may be useful to support discovery mechanisms that use alternative approaches such as registry services [152], hierarchical discovery [30], or federation [143] which can allow better adminstrative configuration over which devices are discoverable.

Thus, one key application of the aggregation mechanism is to provide alternative means of discovery for Obje devices and applications. In addition to allowing devices and services to be deployed onto the network to be discovered "normally" via Zeroconf, it allows Aggregate services to be deployed that allow devices and services that do not support Zeroconf to be discovered as well. In essence, an Aggregate component can act as a bridge between Obje and other discovery protocols. The Obje client first discovers the Aggregate via Zeroconf, and then interacts with the Aggregate to obtain ResultSet granules which in turn support the discovery of components that are known to the Aggregate through some other protocol.

ResultSets embody a great deal of flexibility, and can provide access to a huge range of devices and protocols without requiring that the applications that use them be aware of how they work. In the most simple case, the ResultSet granule can simply forward the operations performed on it by the application to a remote device or service that performs discovery on its behalf. This allows a device or service to serve as a discovery proxy—it can perform discovery using arbitrary other protocols, and return the results to unmodified applications in the form of standard Obje ComponentDescriptors. In other cases, the ResultSet granule may itself provide a custom implementation of a new discovery protocol that executes directly in the client, thereby allowing clients to directly discover other peers that may have been inaccessible to them previously.

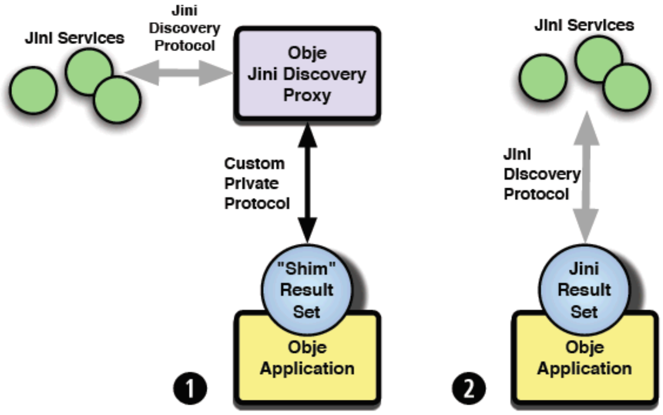


Figure 3-5: Common discovery patterns using ResultSets. On the left, a ResultSet granule provides access to a discovery proxy running on a remote machine. On the right, a ResultSet granule implements a custom discovery protocol that executes in the application.

Figure 3-5 illustrates both of these cases. In the first case in Figure 5, a lightweight “shim” ResultSet communicates using a custom, private protocol to a backend service that serves as a discovery proxy for Jini services; this backend service invokes the Jini discovery protocol [143], and returns discovered services to the client. Because ResultSets leverage mobile code, however, other configurations are possible. For example, a ResultSet granule can provide a full implementation of a new discovery protocol, which can then be delivered to the client where it executes locally. In the second case in Figure 5, the discovery process does not happen in the external service; instead, it happens within the client itself, which has been dynamically extended to use the Jini discovery protocol through the code contained within the granule.

Whether discovery happens in the application or in some external service or devices is up to the service or device that provides the ResultSet granule. In either case, the application simply operates on the ResultSet using the standard iteration operations defined on it, and need not care how the custom implementation does its work.

3.2.2.2.2 Legacy Device Support

The Aggregate pattern is also used by Obje to provide access to legacy devices, meaning both non-Obje devices and devices on non-IP networks. In these cases the Aggregate device acts as a proxy for one or more devices that do not themselves implement the Obje bootstrap protocol. This can allow, for example, USB connected devices on a PC to be exposed as if they were native Obje devices through a "USB Aggregate" that understands how to communicate with these devices, generates ComponentDescriptors for them, and can participate in the Obje bootstrap protocol on their behalf.

Figure 3-6 illustrates a Bluetooth Aggregate. This is a service running on a machine with both a Bluetooth and a traditional wired network interface, that exposes itself as an Obje aggregate device. The ResultSet granule delivered to clients by this component will communicate with the backend machine, where the Bluetooth discovery protocol (SDP) [11] is used to discover devices within range. As devices are discovered, the service generates a ComponentDescriptor for each of the devices, returning them to the client through the ResultSet, as shown in the first illustration. To the client, these aggregate-generated components are indistinguishable from any other component; the Bluetooth aggregate itself simply appears as a collection of all of the Bluetooth devices on the network. Operations on these proxy components, however, are relayed over the wired network to the remote machine, which then uses the Bluetooth RFOBEX protocol to communicate with the device. In this way, aggregates can play a bridging role, acting as an intermediary in interactions between the proxy component for a device, and the device itself. This arrangement can allow the interconnection and use of arbitrary devices, even on different networks.

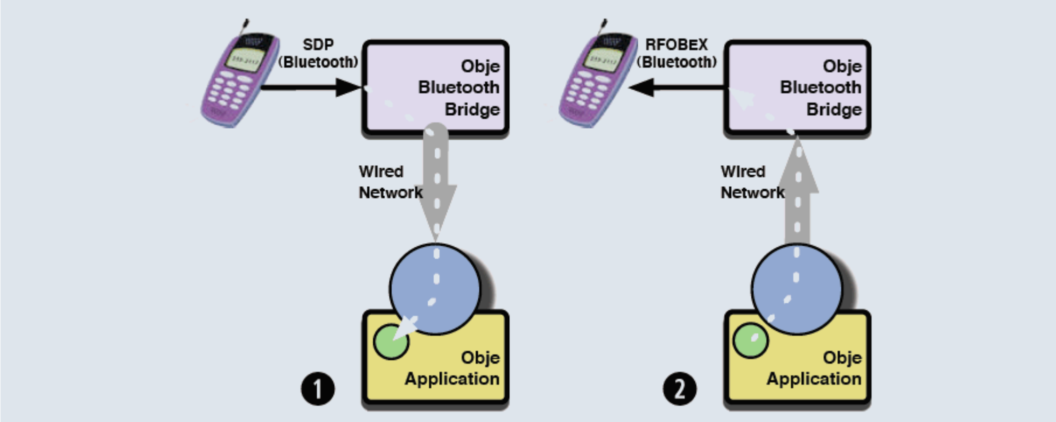


Figure 3-6: Using aggregates to provide access to legacy resources. Here, a Bluetooth bridging service executes on a remote machine to provide access to legacy Bluetooth devices that do not run Obje. As the service discovers a Bluetooth device (on the left), a ComponentDescriptors is generated for it and delivered to the client application via the bridging service's ResultSet granule, embedded in the client. The client can operate on this "proxy" component just as it can any other component. On the right, data transfer operations invoked on the component are forwarded back to the bridge service, which uses the Bluetooth device's native protocols (RF-OBEX) to transfer data to it.

We have created a number of such bridging aggregates for legacy devices, allowing fully networked access to devices such as USB cameras, web cams, and music players, and Bluetooth phones and PDAs.

From a client's perspective, all of these varied uses are possible using the same Aggregate interface, and without rewriting. New discovery protocols can be made available (either directly, or indirectly through proxying) to all clients simply through a single new device on the network. Likewise, bridges to devices residing on different networks, or to devices that do not communicate using Obje at all, can likewise be achieved through the addition of the necessary bridging aggregate to the network. We believe that this range of uses demonstrates the potential of combining fixed interfaces with mobile code—by installing one device on the network, all existing devices and clients can benefit.

3.2.2.3 Metadata

Obje also provides a mechanism that allows devices to provide arbitrary descriptive metadata about themselves. Since our premises dictate that the semantic decisions about when and whether to use a component must ultimately lie with the user, we must provide mechanisms to allow users to understand and make sense of the services available on the network. For example, simply knowing that a device can be a sender or receiver of data provides little utility if no other information can be gleaned about it. For this reason, Obje devices support the ability for applications to retrieve a granule from a device that provides metadata about that device, which may include such details as name, location, administrative domain, status, owner, and so on.

Our representations for metadata are very simple: metadata granules provide access to a simple map of key-value pairs, with keys indicating the names of attributes ("Name," "Location," and so on), and values that are arbitrary objects. The set of keys is extensible, as we do not believe any fixed set is likely to support the needs of all applications or components. Likewise, we do not require nor expect that all applications will know the meaning of all keys, nor share a common metadata ontology. The goal of this mechanism is primarily to allow sensemaking by users, and only secondarily to allow programs to use metadata in their interactions.

3.2.3 Integrated User Control

There are two mechanisms provided by Obje to support user control of devices and services. User can obtain or “pull” user interfaces from services on demand. In addition, services can “push” user interfaces to clients during active communication sessions to allow users to control aspects of the specific session, or to prompt the user for needed information. The “pull” UIs are most useful for situations where the user wishes to configure or manipulate global settings on a device, or wishes to effect controls that are independent of any particular connection. Among other things, this allows Obje to provide access to devices and services that are not oriented around data or media transfer, such as lighting or heating controls. The “push” UIs are expected to be used to allow users to control session-specific parameters such as the playback position of a media stream, or to prompt users to enter needed information such as an access key, confirmation of data destruction, and so forth.

3.2.3.1 On-demand User Interfaces

The first of these capabilities allows client applications to directly request a UI from a device. The ability of a device to provide UIs for controlling them is a kind of “escape hatch,” allowing Obje devices to provide access to functionality that cannot be easily represented using the data transfer or aggregation interfaces. In the case of a printer for example, an application could request the printer’s UI granule and invoke it to display a control panel to the user, allowing control over defaults such as duplex, stapling, and so on.

In keeping with the general requirement to minimize the number of up-front agreements required for interoperation, clients and services agree on the mechanisms for acquiring and displaying UIs, but clients have no knowledge of the particular controls provided by any UI.

Since different client platforms support a wide range of possible UI mechanisms, Obje allows devices to provide multiple UI granules; applications select from these by specifying the platform of the desired UI. For example, a client running on a laptop might request a “javax.swing” GUI, while a PDA might request a “thinlet” XML-based UI. The strategy is flexible in that it allows devices to present arbitrary controls to users, and allows multiple UIs, perhaps specialized for different platforms, for a given device. Regardless of the style of UI selected, Obje provides a message-based communication channel between the UI delivered to the client and the service it is meant to control. This allows a custom control protocol to exist between the control user interface and its host service that is completely opaque to the client application.

One drawback of the approach taken by Obje is that it requires device developers to create separate UI granules for each type of presentation platform. A solution would be to use a device-independent UI representation, such as those proposed by Hodes, et al [63] or UIML [61], and then construct a client-specific instantiation of that UI at runtime. This is an orthogonal consideration to the mechanism for negotiating and delivering the user interface bundles and the construction of the communication channel between the client-hosted UI code and the Obje service. In fact, if a client declared that it could accept UIML (or similar) user interface bundles, and a service declared it could provide one, the current Obje mechanisms would support their interoperation.

3.2.3.2 Transfer Control Extensibility

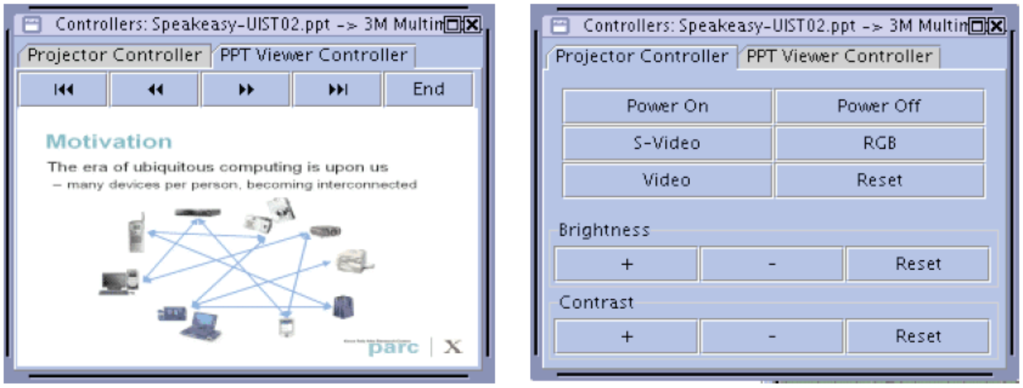
Obje's mechanisms for transfer-specific control allow the user to receive control user interfaces that are relevant to a given data transfer connection that are provided by the components involved in the connection. The granules that are used for transfer control UI delivery are the same as those described for the "pull" UIs, and therefore have the same advantages and drawbacks in terms of client-service interoperability described above.

As a motivating example, if a user uses an Obje client to connect a file of Powerpoint slides to a projector, we would like to display to the user controls specific to that interaction. This may include UIs for the particular model of projector and for the slide show. Of course, we need to be able to do this without requiring that the client be specifically written to understand the details of projectors, or of Powerpoint slides. This ability to acquire and display arbitrary per-component user interfaces is necessary, given the user-in-the-loop philosophy that is fundamental to the Obje approach.

Obje uses the same state notification mechanism defined by session granules to deliver yet another type of granule—which we call controllers—to any of the parties that hold a copy of the session object [111]. Using this mechanism, any party that holds the session can "add" a controller granule, which causes it to be delivered to any other parties holding the session that have solicited interest in receiving such granules; this arrangement allows user interface code to be delivered asynchronously over the network, at the time it is needed, and for presentation by whichever client is managing user interaction.

A key advantage of this approach is that controllers can be added by any party that holds the session, in the same way that any party that holds the session can also update its state. This means that sinks, sources, and typehandlers can all add controllers. For example, in our Powerpoint case, when a connection is established to the projector, the projector (sink) component may add controls for adjusting brightness and other projector parameters. The typehandler—which is the only party in this scenario that would understand Powerpoint data—would add the slideshow controls. Both would be transmitted over the network to the application that initiated the connection.

Figure 3-7 shows a set of simple controllers from the Powerpoint-to-projector example, running on a PDA. Note that the controller delivered to the PDA can provide potentially arbitrary functionality, based on the code that is delivered to it. In this case, the Powerpoint controller displays miniature versions of the current slide on the PDA screen, even though the PDA does not know about Powerpoint, and does not even have the application installed. This controller also allows the user to draw on the slide using a stylus; the strokes are communicated back to the Powerpoint typehandler executing on the projector, where they are rendered over the slide. Meanwhile, the Projector controller allows the user to control aspects of the projector's operation, such as the brightness, contrast, and input port. Because this particular PDA client is running on a device with limited screen space, it displays each received controller in a separate tab that can be toggled between by the user. If the client were on a larger device, it might instead decide to lay out both controllers on a single screen, providing users with the ability to easily control all parameters of the current session in one place.



Motivation

The era of ubiquitous computing is upon us

– many devices per person, becoming interconnected

Figure 3-7: Projector and Slide Show controllers running on an iPaq PDA. The client application solicits to receive controllers from any session in which it is involved. Received controllers for a given session are presented in tabbed panels.

3.2.3.3 Other Applications of "Push" UIs

While "push" UIs are only supported within the context of data transfers in Obje, the concept could be extended to other situations. For example, an Aggregate component may wish to provide an alternative UI for browsing its contents, or even portions of its contents, rather than allow the client to use the generic Aggregate service interface to retrieve and organize its contents. This could be desirable especially if the Aggregate contains a specialized type of data (e.g., music or movies) for which an optimized browsing experience might be of great benefit to the user. Such a user interface could be "pushed" to the user after the client has established the connection with the Aggregate via the ResultSet mechanism. Since many Aggregates will likely be content with the default browsing experience, and some will be content only under some circumstances, it would be beneficial to allow the Aggregate to communicate the availability of (or "push") a customized UI only when necessary, rather than expect the client to "pull" a specialized Aggregate UI in all cases.

3.3 Experiences with the Obje Framework

Over the course of six years (2000-2006), a number of applications were built atop the Obje Framework. I will describe several of them briefly in this subsection and also introduce the two applications that will be the subject of the next two chapters: the Obje Display Mirror and OSCAR. Each of these applications was designed to work with Obje services that they discovered on the local network. Thus, in addition to developing applications, I and my collaborators developed approximately two dozen Obje components including components for projectors, displays, speakers, microphones, webcams, "file spaces" (shared directories), screen mirroring (a VNC DataSource), printers, "removable drives" (e.g., digital cameras and portable music players), RSS feeds, and Internet radio stations. Additionally, we created discovery Aggregates for Bluetooth, Jini Discovery, UPnP, Infrared, and HP's eSquirt.

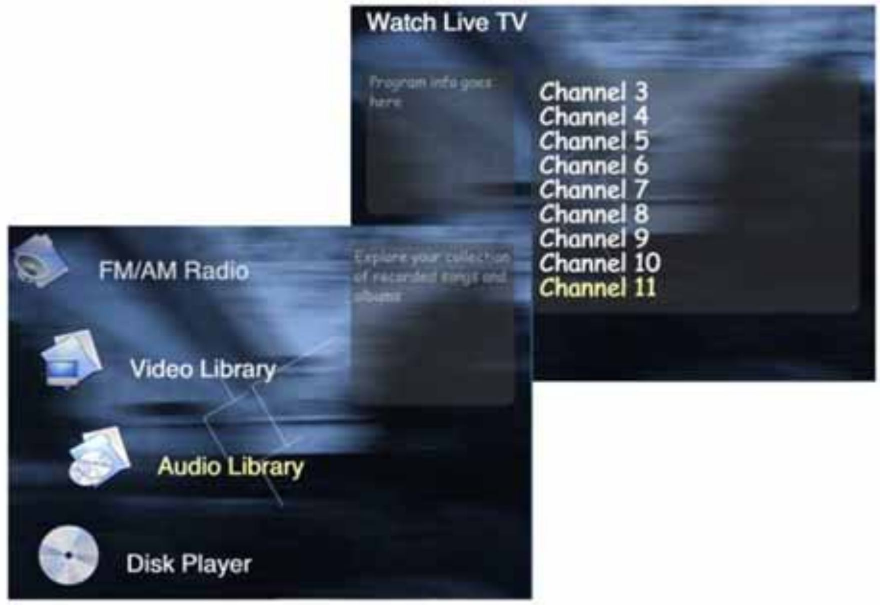
The applications described in this section are split into two coarse categories: specialized applications and general-purpose composition tools. The former category represents applications that were designed to present a "traditional" user experience in the sense that they allow users to perform a (somewhat) constrained set of tasks, and they have certain pre-defined sets of functionality built in. Even these specialized applications take advantage of Obje to provide a high degree of flexibility and adaptability, especially with regard to the easy incorporation of new devices and services that become available. The general-purpose tools all share the characteristic that they allow users to discover, browse, control, and connect *any* Obje components that are available for discovery, and they do not provide any particular functionality above and beyond what is provided by the components with which they interact.

3.3.1 Specialized Applications

Obje applications for two domains are presented in this section: home media consumption and workplace collaboration. In the former category, I describe the Obje Set-Top Box, which is an application that allows users to discover and playback various media through a connected TV or through other networked devices throughout the home. I will also describe three applications that were developed to support collaboration. Casca and the Sharing Palette provide generalized sharing of files and devices among groups of collaborators. The Obje Display Mirror supports the easy sharing of information through the use of shared public displays.

3.3.1.1 The Obje Set-Top Box

The Obje set-top box is a specialized application for media-oriented applications [43]. This system allows users to interconnect audio-and video-related services hosted on the “box” (in our prototype implementation represented by a dedicated computer running the Obje set-top box software), as well as Obje devices and legacy devices elsewhere on the home network. By making assumptions about the context of use (storing and playback of media files), the interface can be streamlined somewhat: the system discovers and groups all devices and content available on the home network into either “audio” or “video” classes (depending on the MIME types they support), and organizes these into menus. Selecting a media source brings up a dialog to either play the content "here" (meaning to one of the sink devices connected directly to the set-top), or to a list of compatible sinks discovered elsewhere on the network. Figure 3-8 shows two screens from the system.



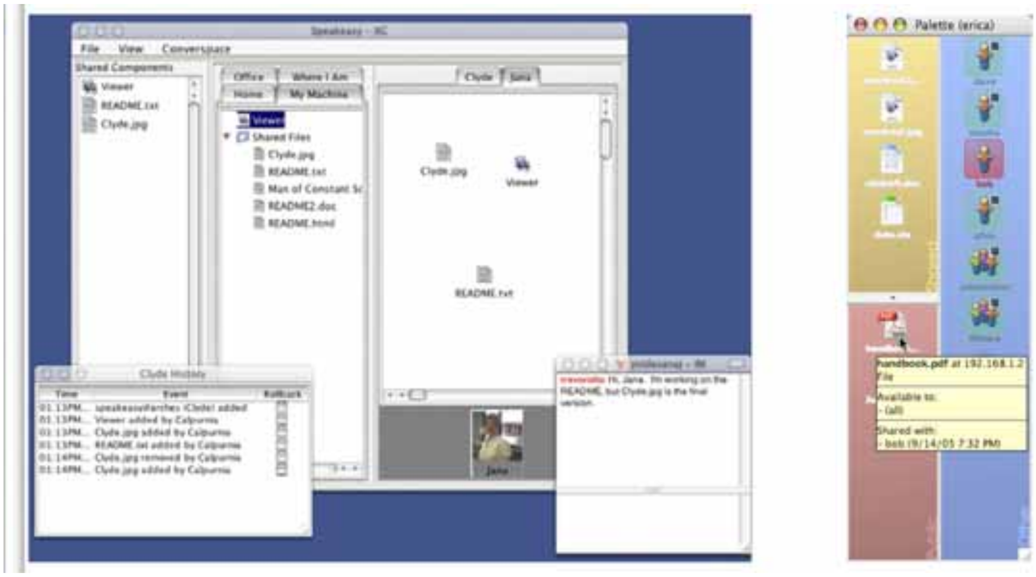
Program info goes here

Explore your collection of recorded songs and albums

Figure 3-8: Two main screens from the Obje set-top application. The image on the lower left shows some of the main menu items in the user interface. Each selection represents a built in template of functionality related to a particular media or device type. The image on the upper right shows a specific user interface for tuning channels on a television tuner component.

3.3.1.2 Obje Collaboration Tools: Casca and the Sharing Palette

One area I and my collaborators explored in some depth is the potential of interoperability frameworks such as Obje to support easier collaboration among users, including not only information sharing, but also device sharing. One such tool we created to support ongoing, small-group sharing is called Casca [41]. This tool was designed to allow users to publish access to files and devices from their laptops into a shared space, thus making them available to others who are "members" of that space. Another application that uses Obje to support collaboration is the Sharing Palette [157], which provides the ability to push content to collaborators using a lightweight icon bar, as well as to establish sharing groups and publicly-accessible files and devices. Both of these tools leverage Obje to allow collaborators to share not only files, but also access to devices and services. For example, these tools can allow a user to provide access to his or her webcam, a restricted filespace, or home printer to a collaborator: essentially, whatever Obje devices or services are available can be shared using these tools. Figure 3-9 shows both of these.



Time	Event	Rollback
01:13PM...	speakeasy/arches (Clyde) added	
01:13PM...	Viewer added by Calpurnia	
01:13PM...	Clyde.jpg added by Calpurnia	
01:13PM...	README.txt added by Calpurnia	
01:14PM...	Clyde.jpg removed by Calpurnia	
01:14PM...	Clyde.jpg added by Calpurnia	

[illegible]: Hi, Jana. I'm working on the README, but Clyde.jpg is the final version.

handbook.pdf at 192.168.1.2
File
Available to:
- (all)
Shared with:
- bob (9/14/05 7:32 PM)

(a) (b)

Figure 3-9: Casca and the Sharing Palette are two collaborative applications built atop the Obje Framework. The Casca main window ((a), in the back) is divided into three panes, with the leftmost pane showing all devices and services being shared by the current user, the center pane showing all discoverable devices and services, and the rightmost pane showing the contents of a single "Converspace." The Converspace is represented by an initially blank canvas into which files, services, and devices can be dragged and dropped, sharing them with the other members of that shared space, who are shown along the bottom right. The Sharing Palette (b), a lightweight interface for small group sharing. It, too, is divided into three regions, with the top left showing all resources that are shared with particular individuals or groups, the bottom left showing all resources that are publicly available to all users, and the right showing all other users and groups with whom sharing is possible.

3.3.1.3 The Obje Display Mirror

The Obje Display Mirror (ODM) [110], shown in Figure 3-10, allows easy networked discovery and control of displays including projectors and plasma screens from users' laptops. This application only discovers devices that can accept viewable media types; in our environment, this typically includes devices such as projectors and plasma displays. Users can select a discovered display to mirror their laptop screens onto it. The user interface limits the complexity seen by the user by restricting the user's view onto the network of only appropriate peers. The ODM was in daily use around PARC, in three meeting rooms, for approximately one year, and was intensively observed in one of those meeting rooms for a period of six months. The Obje Display Mirror, along with the study of its use and adoption, is described in greater detail in Chapter 4.

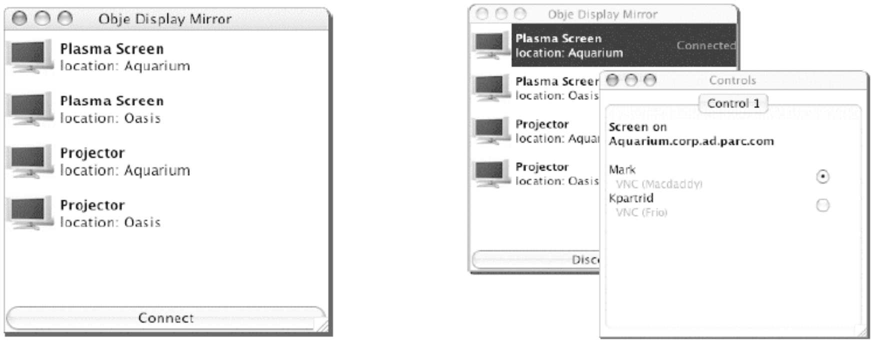


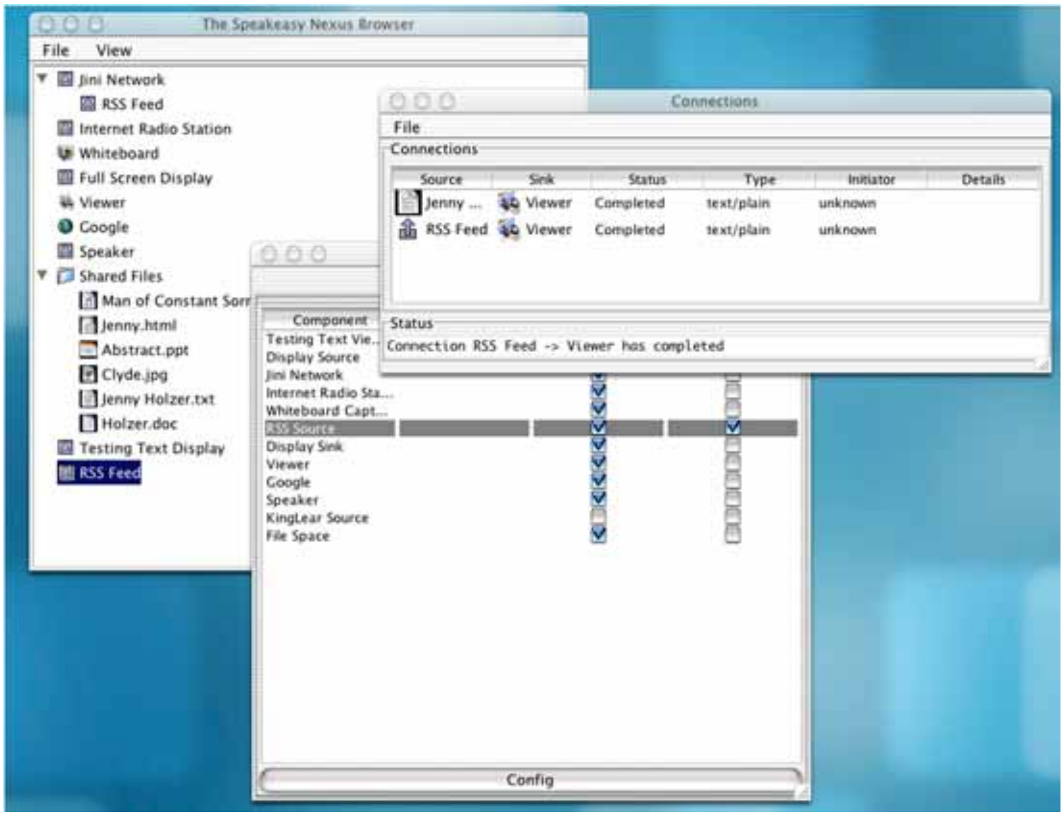
Figure 3-10: The Obje Display Mirror before and after a connection is made. On the left, the user sees a list of available screens. After connection, a control UI is presented to all connected users of the selected screen, allowing control of the display to be shared.

3.3.2 Generic Tools for Service Composition

Providing powerful, end user-oriented tools for service composition will become even more essential as we move to a world in which device connectivity is not limited by what the developers of those devices foresaw, but by users' desires and needs. In addition to the special-purpose applications described above, a number of general-purpose tools have been built on top of the Obje Framework.

3.3.2.1 Generic Browsers: Nexus, Wander, and the Orbital Browser

Some of the earliest applications our team built were generic browser applications, which provide direct access to the Obje devices on the network. One of these applications, the Nexus, is shown in Figure 3-11. The Nexus and its successor Wander (not shown, but visually and operationally very similar to the Nexus) present simple lists of discovered devices, and map user operations in the UI onto Obje device operations: for example, dragging and dropping a source device onto a sink initiates a data transfer; double clicking an aggregate device "opens" it, revealing the components to which it provides access. The Orbital Browser [37](Figure 3-12) provides a lightweight interface that uses only a knob as an input device to select and compose services. Rotating the knob navigates a cursor through the available set of components, while depressing the knob selects the currently highlighted component. Holding the knob down for a longer period of time expands or contracts aggregate components. Through combinations of these actions, users can get information about each component, and can make and break connections between them. This tool was intended to demonstrate a minimalist interface for providing open-ended service composition in the sense that it can be operated with limited input capabilities (two operations—move and select—are required, a third operator allows the user to move in two directions for more efficient navigation), and used well with a range of displays from very small to very large.



Source	Sink	Status	Type	Initiator	Details
Jenny ...	Viewer	Completed	text/plain	unknown	
RSS Feed	Viewer	Completed	text/plain	unknown	

Connection RSS Feed -> Viewer has completed

Figure 3-11:The Nexus is an example of generic service browsers that allow a user to discover, connect, and control Obje components available on the network.

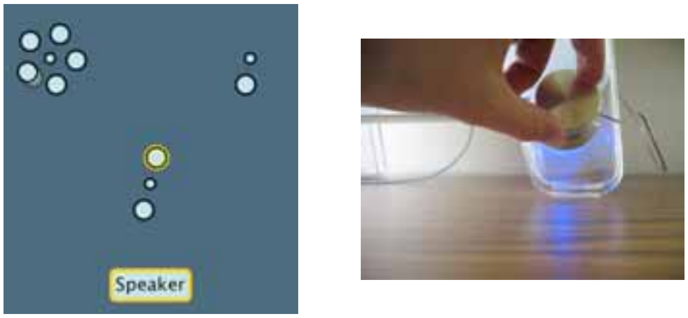
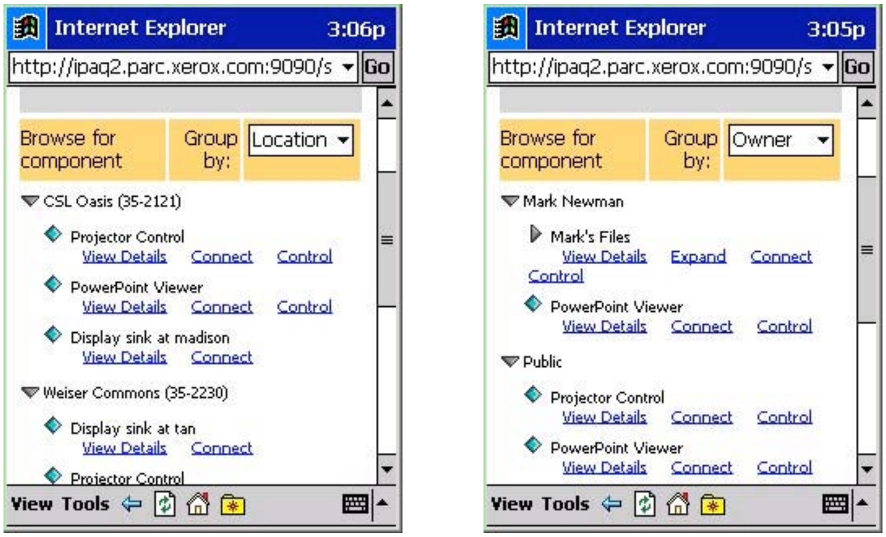


Figure 3-12: The Orbital Browser's interface, and its input device. Spinning and pressing the knob allows users to navigate to any discoverable device, initiate and terminate connections, and so forth.

Though these UIs were not tested with end-users, experiences with these applications reinforced our belief that such generic applications, which allow users to discover and interact with arbitrary devices in an ad hoc fashion, will play an important role in any ubiquitous computing future. In the absence of specialized applications for every conceivable task, a more generic tool—one without specialized domain knowledge —will necessarily play a part. However, it remained unclear how to present these functions to end users who may have difficulty understanding the rich functionality of which their networks of devices are capable.

3.3.2.2 The Speakeasy Handheld Browser

We have built and studied a number of these applications. The first (reported in [40, 112] and shown in Figure 13), was a web-based application designed for access on a PDA. Our early experiences with this tool informed a number of our architectural design choices, particularly in our data transfer design. For example, early versions of Obje relied on external filter services, which could be chained when performing a data transfer. The UI designs that were implied by this mechanism required the user to "hand assemble" chains of data transformations to connect two devices in situations where their data types would have otherwise been incompatible. While other sorts of user interfaces could have been created—for example, by automatically finding paths of filters—such arrangements remove the user from the process of determining which filter services to use, which is important in the situation where filters execute on the network, on hosts with different speeds, or different levels of trustworthiness. The user, not the system, is better equipped to consider these factors.



Internet Explorer 3:06p

http://ipaq2.parc.xerox.com:9090/s Go

Browse for component Group by: Location

▼ CSL Oasis (35-2121)

- Projector Control
  View Details Connect Control
- PowerPoint Viewer
  View Details Connect Control
- Display sink at madison
  View Details Connect

▼ Weiser Commons (35-2230)

- Display sink at tan
  View Details Connect
- Projector Control

View Tools

(a)

Internet Explorer 3:05p

http://ipaq2.parc.xerox.com:9090/s Go

Browse for component Group by: Owner

▼ Mark Newman

- ▶ Mark's Files
  View Details Expand Connect Control
- PowerPoint Viewer
  View Details Connect Control

▼ Public

- Projector Control
  View Details Connect Control
- PowerPoint Viewer
  View Details Connect Control

View Tools

(b)

Figure 3-13: The Obje Handheld Browser running in a PDA web browser. Basic facilities are shown for sorting and selecting components by location (a) and by owner (b).

We believe that one approach to resolving this dilemma is to create facilities for moving from device-oriented interactions to task-oriented ones. That is, rather than specifying the various data flows among individual devices, the user would simply select a desired task and the system would instantiate the necessary device-to-device interactions. Of course, actionable representations of such tasks must exist for this option to be available. There are a number of ways such representations could come into being. They could be created by experienced users, for example, much in the way that Excel macros for a given site are often created by local "gurus," and then shared throughout that site [94]. We have explored this approach, which we call "task-oriented templates." These templates contain slots that describe the devices that are needed for a given task, in terms of input and output types, metadata properties, and so forth. When a template is instantiated, the template engine attempts to match available devices to slot descriptions (perhaps with help from the user), and then creates the necessary connections among them. For example, a "give a presentation" template would specify slots for projectors, speakers, and source slides file; instantiating this template would create the connections among Obje devices to allow easy setup of a conference room.

3.3.2.3 OSCAR

OSCAR is a direct manipulation touchscreen-based interface for end-user service composition in a home media network. Drawing on the experiences of the Handheld Browser, OSCAR (for Obje Service Composer and Recomposer) uses templates as the foremost feature in its interface, and allows users to create new setups as well as use and adapt existing ones. As I shall discuss in detail in Chapters 5 and 6, OSCAR presents templates as "setups" that they can instantiate to control and interact with networked media appliances around their homes. In some sense, OSCAR represents the culmination of all of the previous experiences with applications in Obje, providing the ability to perform ad hoc integration of arbitrary components while providing a mechanism for delivering specialized application semantics in a way that does not compromise future interoperability or the ability of users to create a customized, integrated user experience of their environments.

3.4 Discussion

Before concluding this chapter I will present a summary of Obje's key contributions, followed by a discussion of the tension between specialized applications and generic tools in ubiquitous computing, given the foregoing discussion of the need for ad-hoc interoperability in evolving, dynamic environments of networked devices and services. This latter discussion will lay the groundwork for the following chapters that discuss two representative Obje applications in greater detail: the Obje Display Mirror—a specialized application supporting the use of public displays in a workplace environment—, and OSCAR—a generic tool for end-user composition in home media networks.

3.4.1 Contributions of the Obje Framework

The Obje Framework provides a novel approach to interoperability that is based on establishing a minimal set of agreements up front (at development time) and allowing new interoperation functionality to be distributed via mobile code among cooperating entities at runtime (after deployment). This approach, called the

Recombinant Computing approach [42], is designed to support robust interoperability among devices and services that do not have to be specifically designed to work together. In addition to allowing decoupled development of reusable services, such an approach is more robust than conventional approaches to evolution of standards for data transfer protocols, media encodings, discovery protocols, and so forth. In addition to presenting arguments for why such an approach offers improvements over alternatives, I described a number of services and applications that had been built atop Obje [41, 43, 110]. These examples showed not only the range of functionality that can be delivered via Obje's mechanisms, but also the reusability and recombinability of services among each other to support multiple end-user applications. The interoperability provided by Obje is what enables such range and reusability.

Another novel feature of Obje is the ability of services to *push* user interfaces to clients at various points during an active data transfer session. Many distributed service frameworks (e.g., Jini [158], UPnP [153]) allow clients to explicitly request, or *pull*, user interfaces from services. These user interfaces are important and useful, and indeed they are also supported by Obje. The patterns and mechanics that support this "push" functionality are a novel feature of Obje and they allow clients to be written in a general way without requiring that they know the particular details of any particular service's data transfer semantics.

Obje is designed to support end-user composition. This means that Obje specifies particular service interfaces that define the *roles* that services can play in certain interactions. Many ubicomp and distributed service frameworks stop short of specifying particular service interfaces (e.g., Jini [158], iROS [76], Gaia [131], one.world [54]), preferring to provide basic mechanisms that can be adapted to a variety of different styles of functional decomposition into services. The notable exception to this trend is the UPnP [153] standardization effort led by the Digital Living Network Alliance (DLNA) [35], which shares with Obje a complete specification of service interfaces. Specification of service interfaces is required for end-user composition, because end-user composition tools must be programmed to address the set of service types that they expect to encounter. Unlike DLNA/UPnP, Obje provides support for extensible structured metadata attached to its services, which is intended to help users to make sense of a service by viewing its attributes such as location, access policies, ownership, functional capabilities, and other arbitrary information. DLNA is primarily aimed at supporting application and device developers, and as such has not prioritized features that would support end-user sensemaking and composition.

3.4.2 Discussion of Specialized and Generic Application Types

There is an inherent tension with creating usable interfaces for a world of ad hoc interoperability. Generic tools—meaning, tools that can work with arbitrary devices—are necessary to take full advantage of the infrastructure. And yet, such generic tools by their nature do not have the tight semantic integration — the “understanding” of what certain devices or content types “mean,” built into their programming—that support good ease of use. Further, it requires that the underlying mechanisms provided by the infrastructure be understandable and usable by users if they are to take advantage of this power.

Interfaces that directly expose low-level operations are difficult for users to work with. The experiences with the Speakeasy Handheld Browser showed that the basic concept of exposing data flows between devices as a first class operation in the interface was confusing to many users. During user testing for the Handheld Browser, for example, some users wanted to simply “open” a Powerpoint file once it had been located, expecting it to appear on the display in the room. These desires seem to betray expectations based on PC use, in which the tight coupling (both semantically and physically) between the PC and its display device allow actions such as opening a file to have a relatively unambiguous meaning. In a more loosely coupled world, potential ambiguities exist, which must be resolved by the user.

On the other hand, specialized applications built atop Obje and leveraging its support for ad hoc discovery, robust interoperability, and dynamically distributable user interfaces, point to the power of allowing flexible yet tailored device access through traditional applications. Such applications may fit particular users' needs better, and may help constrain the set of available options, thus making the particular operations they support easier to use for many users. The drawback, clearly, is that specialized applications need to be written for each anticipated usage scenario, and this becomes increasingly challenging as the environments, devices, and desired tasks with which users are engaged continue to multiply.

Ultimately, the sorts of open-ended device use available under systems that support ad hoc interoperability point to the importance of service composition techniques. Service composition is the process of combining multiple, discrete services or devices together to achieve some higher-level goal. In Obje, I and my colleagues have explored a number of different approaches to service composition, incorporating different user interfaces styles (e.g., the Orbital Browser, the Handheld Browser, and OSCAR) and temporal framings of service composition, ranging from ephemeral, one-off browsing and connection to persistent, reusable "templates" for compositions that are created, adapted, used and perhaps ultimately shared by end-users.

3.5 Summary

The goal of the Obje Framework, as laid out at the beginning of this chapter, is threefold: to enable seamless, robust, evolution-proof interoperability; provide ubiquitous and integrated user control of multiple networked devices; and to lay the groundwork for end-user composition. I have described how the Obje Framework implements the notion of recombinant computing, especially with regard to robust and extensible interoperability along the dimensions of data transfer protocols, data types, discovery protocols, physical networks, and user interfaces. I have also described how Obje allows services to provide two kinds of important user interfaces to clients without requiring that the clients have any advance knowledge of the services' capabilities, interfaces, functions, etc. I have also showed how Obje can be used to support both speciablized applications and generalized tools that support end-user composition.

In the next chapters, I will describe experiences with two applications built atop Obje. The first is the Obje Display Mirror (Chapter 4), which is a narrow, simple (from the user's perspective) application that provides wireless access from a user's laptop to any shared display (e.g., projector, plasma screen, computer monitor) in a workplace with a minimum of interaction overhead. This experience reveals some of the challenges of deploying, maintaining, and evaluating even a "simple" networked service over an extended period of time. The second application I will describe is OSCAR (Chapter 5), which is a direct manipulation touchscreen-based interface for end-user service composition in a home media network. The OSCAR experience, along with its user study (presented in Chapter 6) shows first of all that ad-hoc interoperability and end-user composition can be presented to non-technical users in a way that is palatable, engaging, easy-to-learn, easy-to-use, and regarded as useful; and second of all that such benefits can be gained only when details about the usability and conceptual framing are carefully considered.

4 The Obje Display Mirror

As described in the latter part of Chapter 3, a number of applications have been built atop the Obje Framework in order to understand and gain experience with its various capabilities and aspects. One such application, the Obje Display Mirror, was significant in that it was the only Obje application that received sustained, daily use by individuals outside the project team. In this chapter, I will present the initial goals and motivation for the Display Mirror project[4]; the initial study of workplace device usage that led to the design of the Display Mirror; the implementation and deployment of the Display Mirror; and the continued study of workplace device usage after the deployment and adoption of the Display Mirror that showed that small but important shifts in workplace practice and user experience had taken place as a result of the new technology.

[4] The work presented in this chapter was principally conducted by me, but benefited by contributions from Nicolas Duchenaut, Keith Edwards, Trevor Smith, and Jana Sedivy. Portions of the work presented in this chapter have been previously described in [110].

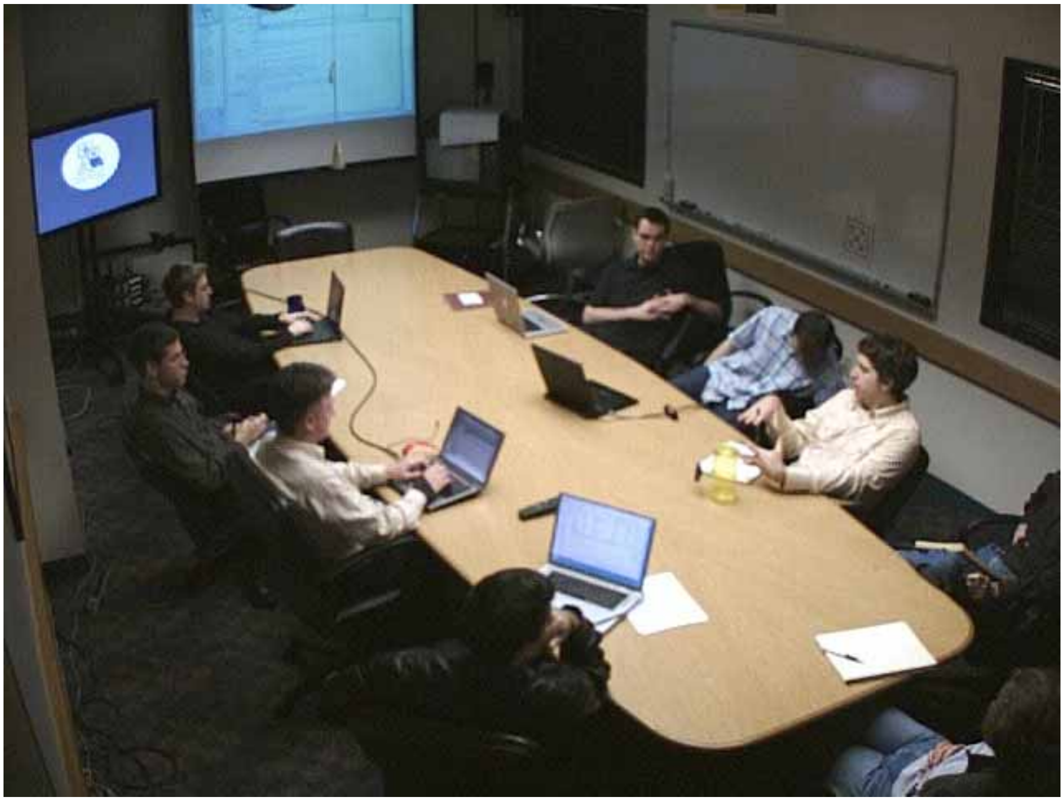


Figure 4-1: The Obje Display Mirror was used in meetings on a regular basis over a period of approximately six months. This figure shows the type of meeting for which the Obje Display Mirror was useful.

In some ways, the scope of the project and the results were not what was expected at the outset, and so I will first present a bit of background and context regarding the initial project goals and how those shifted during the project's course.

1.1 Initial Goals and Context

The goal of the work presented in this chapter was to create an Obje-based environment for a group of people to use for an extended period of time in order to see if and how the capabilities afforded by Obje's interoperability and flexibility would affect everyday practice. Towards this goal, I, along with my colleagues undertook to instrument the work environment of the PARC Computer Science Lab (CSL) with Obje services and devices, and to provide client programs that would allow any of the members of CSL to access and control those services. This work followed on the Handheld Browser (HHB) work discussed in the previous chapter, which initially had shared the same goals of creating a persistent, usable environment of services with which users could interact on a daily basis. As became clear during the development of the HHB, however, the Obje software was not yet robust enough to support a sustained deployment with dozens of users. Due in large part to the HHB experience, I and one other team member undertook to rebuild the Obje infrastructure from scratch during 2003-2004, and our efforts resulted in a platform that was significantly more robust in terms of development of services and clients, deployment, maintenance, and use. As I shall discuss later, we still encountered significant challenges along these lines as the Display Mirror progressed, but we were able to get much farther than had been possible with previous versions of Obje.

The accomplishments of the Display Mirror project were somewhat more modest than the original goals. Where we had set out to deploy dozens of installations of perhaps 5-10 different types of services, we ended up deploying 4-6 installations of two services, the Obje Display Mirror and the Obje Whiteboard Capture. Of these, only the Display Mirror was deployed over a long period of time and received regular use by users outside the project team.

The reasons for this scaling back were several: the resources and effort required to sustain the various service installations and, more significantly, the client software, were more substantial than anticipated; the burdens of promotion and social pressure required to sustain interest in and adoption of the technology, especially in the face of periodic technical failures, were also great; and, finally, these realizations of greater resource requirements combined with shifts in organizational priorities and funding sources for the project resulted in pressure on the project team to seek other outlets for Obje technology outside the workplace before the grander goals of the project could be realized.

Nevertheless, the accomplishments that were realized on the project were significant, and in some areas were greater than we might have hoped during the initial formulation of the project. In particular, we made an early decision to build tools for assessment and evaluation before deploying any new technology, so that we could compare users' behavior before and after the introduction of the technology. The focus of the project then became as much about the design and evaluation of ubicomp technologies as it was about studying the effects of introducing Obje technology into an environment.

An initial meta-goal of the project was to introduce new technologies and capabilities into existing practice in as non-disruptive a manner as possible. By doing so, I believed that we could interleave the new capabilities into existing practice more easily, and that we could more accurately assess the effects of introducing each new technical component on practice. For reasons that I will return to in the discussion, I now believe this approach of non-disruption was misguided. Even small technical interventions are disruptive by their very nature, and confronting this fact directly may be more productive than attempting to minimize it.

The starting point of the project, therefore, was the goal of introducing technology for long-term use, and attempting to do so in a gentle, "unremarkable" way. Thus I will begin the discussion of the project with a discussion of Tolmie, et al.'s "Unremarkable Computing" [148] and show how the agenda presented in that work informed our thinking about our own project.

4.1 Supporting Unremarkable Computing

In their paper "Unremarkable Computing," Tolmie, et al. encouraged designers of ubiquitous computing technology to investigate the use of computing to support the mundane, everyday routines of life [148]. This agenda lines up with the original vision of ubiquitous and calm computing as expressed by Mark Weiser [160] and refined via ideas like "everyday" [107] and "ambient" [163] computing that have driven much research over the past several years.

The work presented in this chapter is informed by a broader methodological and theoretical framework from the social sciences. Ethnomethodology [49] alerts us to the importance of studying the "detailed and observable practices that make up the incarnate production of ordinary social facts" [92]. These "ethnomethods" are, by definition, mundane and invisible (which is why they require such detailed observation to be uncovered or, alternatively, some form of "breaching experiment" to foreground them). Yet they are the foundations of the orderly functioning of society. Along the same lines, Suchman's influential book *Plans And Situated Actions* [141] emphasized the situated nature of human activity, and the importance of the context surrounding any instance of technology use. This context is, at first sight, also mundane and invisible. Thorough studies of, for instance, a person's work environment, are meant to reveal its effects.

In this chapter I describe the attempts made by me and my collaborators at studying, designing, and deploying ubiquitous computing technology specifically meant to support mundane practices. There are two sides to the mundane that are important for systems designers. On the one hand, achieving mundanity is a *goal* we have for much of the technology we design—we hope that our systems will eventually integrate so well into our users' lives that they will become ordinary and commonplace. At the same time, mundanity can be seen as a *topic for investigation* in the design of ubicomp technologies. In this latter case, we set out to understand, support, and improve activities that have *already become mundane*. In this paper, we hope to shed light on this latter, less studied aspect of the relationship between technology and the mundane.

To illustrate our focus on the application of ubicomp technologies to the mundane, we report on the design and deployment of a lightweight service to support and enhance the commonplace activity of connecting a portable computer to a shared display (e.g., VGA projector). Our experiences with this service provide insight into the strategies and methods that are required to tackle this peculiar design domain. However, our investigations into the unremarkable were not without difficulties. We encountered important challenges before, during, and after the deployment of our mundane technology. These challenges either differ entirely from those traditionally encountered during application design, or are amplified versions of more traditional problems. I discuss these difficulties, with the hope of informing and facilitating future work in this domain.

In the coming pages, I first describe our initial, six-month study into device-oriented behaviors in a meeting room at our facility. Based on this study, we designed and deployed a service, the "Display Mirror," that was deployed in the same meeting room, and describe our observations regarding the adoption and use of the service after six months of (mostly) continuous availability. I then discuss what we learned from our experiences in the design, deployment, and evaluation of technology to support mundane activity. Finally, I provide recommendations for other researchers interested in supporting unremarkable, everyday behavior.

4.2 Observing Device-Oriented Practices in the Workplace

The intent of our work was to advance the agenda of unremarkable computing by not only describing people's interactions with technology but by introducing technology that intervenes in those interactions, and by understanding the results of doing so. Thus we found it necessary to augment the ethnographic methods employed by Tolmie, et al. with quantitative data collection via longitudinal sampling in an effort to provide concrete information about the nature and quantity of various interactions that were taking place, and to provide a before-and-after comparison of behavior among our study population.

4.2.1 Studying Meeting Rooms

We focused our observations on one meeting room[5] in our facility that is used regularly by a variety of individuals from different parts of the organization including researchers, managers, and support staff. It is also commonly used to meet with external visitors. The room is equipped with a large oval conference

[5] More precisely, we deployed the Display Mirror in three meeting rooms and two public areas dedicated to informal congregation, and we initially tracked before-and-after device usage in two meeting of the rooms. We stopped tracking usage in the second meeting room once we realized that its usage was dominated by members of our own project team. Thereafter we came to regard the second meeting room as a pilot testing area and focused our observations on the first room. All data reported in this chapter was collected from the first (non-pilot) meeting room. The third meeting room was added late in the deployment, and was an executive conference room used exclusively by PARC's senior staff. Instrumenting and observing in this room was not permitted. The Display Mirror installations in the informal public congregation areas received little or no use, in large part because people were not accustomed to using laptops in these areas.

table (14 seats), two whiteboards, a whiteboard capture camera, wireless and wired network connections, and a ceiling-mounted XGA projector.

Meeting rooms have been extensively studied in the CSCW literature [118, 151]. They have also been privileged experimental sites for ubicomp researchers [75, 140]. The reasons that CSCW and Ubicomp researchers have paid so much attention to meeting rooms is clear: they are sites that are rich in human, device, and computer interaction. Moreover, a less frequently acknowledged property of meeting rooms is that they are "boring"—that is, they are the locus of many activities that are now considered mundane, such as presenting information to a group or collectively reviewing documents. Our interest in meeting rooms stems in part from this combination of technological richness and mundane activity.

Setting out, a particular goal in undertaking this project was to determine how the presence of physically situated, continuously available networked services such as screens, speakers, printers, audio and video capture devices, and so forth might influence the expectations and practices of users who were exposed to them over a long enough period to allow them to adopt them into everyday practice. Before designing any specific technology however, we had to understand current practices. We therefore designed a study of existing device-oriented behaviors in meeting rooms with three goals in mind: (1) to determine what types of activities were taking place during meetings that could be helped or improved by the addition of continuously available services; (2) to determine which devices and resources available in the meeting rooms would be most useful if made available as networked services; and (3) to learn what types of portable personal devices were being brought into the room that we could leverage as client devices as well as a source of additional services that could be combined with the ones embedded in the room.

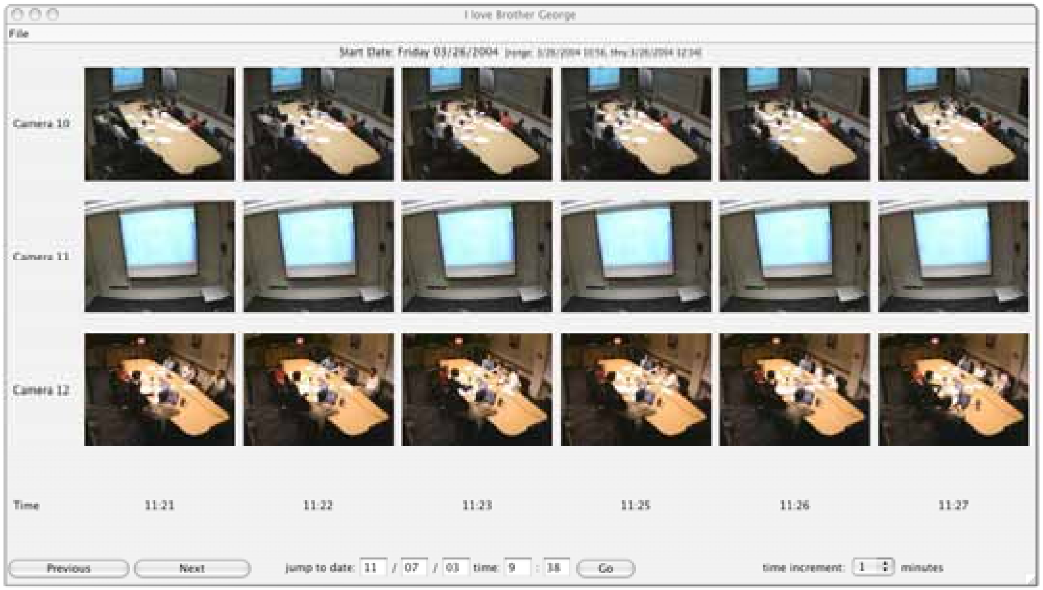
We describe our study and the methods we used below. We then look at how the results of this study informed the design of one of our services: the Display Mirror, which allows users to connect to shared meeting room displays (e.g., projectors and plasma screens) without using cables.

4.2.2 Observation Methods

Our goals for this project made traditional ethnographic observations alone insufficient. Indeed, we needed to capture and analyze longitudinal data: the sporadic, interleaved and fine-grained nature of the activities we wanted to observe (e.g., connecting a laptop to a screen) made it impossible for us to dedicate the number of person-hours needed to have researchers present in meeting rooms continuously. These constraints led us to consider techniques that sample behavior or activity, rather than ones that demand constant attention. The classic sampling technique, the Experience Sampling Method (ESM) [28] has been used for a variety of purposes related to ubicomp [26, 48, 68, 73]. This technique did not seem to be appropriate for us to use, because it depends on the subjects being able to articulate the relevant aspects of their experience at the time they are sampled. Since a portion of the activity we wished to study is often not the conscious aspects, this would not work.

Another related technique is Lag Sequential Analysis (LSA), which was applied to the design and evaluation of LabScape [25] in a way that closely resembled our approach. In LSA, a set of temporal interactions are captured and analyzed, either using video or in-person observations. The interactions are coded at the granularity of the *lag*, a short time duration (in the LabScape study, the lag was one minute). A set of interesting event types are chosen, and for each lag the presence or absence of the event is noted. This provides a record of all significant events at the granularity of the lag. Most importantly, it provides a record of temporal relationships among events (e.g., precedence, co-occurrence, etc.) However, since we were seeking to capture and analyze a greater volume of data—spanning months rather than the hours or perhaps days that are conventionally covered by LSA—we had to find a way to meet our needs with a considerably smaller investment in terms of capture and coding effort.

Therefore, we elected instead to take snapshots of meeting rooms and their visitors once a minute from three angles. It turned out that this low sample rate of data (as compared to, say, streaming video), along with the assurance that we were capturing using low-resolution consumer webcams (640x480 resolution) and were not capturing any audio, was helpful in assuaging our users' concerns about loss of privacy. Data representing times when people were in the room was later coded using a tool that we built especially for this purpose (see Figure 4-2). We supplemented these automated and coded observations with interviews and direct observations of meetings.



Start Date: Friday 03/26/2004 [range: 3/26/2004 10:56 thru 3/26/2004 12:34]

jump to date: 11 / 07 / 03 time: 9 : 38

time increment: 1 minutes

Figure 4-2: We developed a custom tool for reviewing and coding our observational data. Each column represents a time sample, and each row represents one of the cameras installed in the room.

4.2.3 Initial Study Results and Observations

The results of our study are divided into two phases, which are distinguished by the technology that was available in the meeting room at the time. The first phase (Phase 1a) lasted two months (February and March, 2004) and represents the technology that was available before we began our study, as was described earlier. The second phase (Phase 1b) lasted almost five months (April through August, 2004) though the data analyzed in this chapter covers only two of those months (mid-May through mid-July, 2004) .Phase 1b represents the period after we added a 50” plasma display and a public PC with wireless mouse and keyboard. The PC was connected to the internal network but left logged in using a local account so that anyone in the room could gain access to the Internet and those with access privileges on the corporate network could use the PC to log into their own account, access their home directories, check their email, etc. The PC, projector, plasma screen, and one additional VGA input cable were all connected together with a 2x2 VGA matrix switch that allowed either source (PC or VGA input) to be directed to either or both of the displays (plasma or projector). Our goal with the introduction of these additional devices was to add additional display options and client capabilities using the best off-the-shelf technology we could find.

We analyzed 47 meetings from Phase 1 in detail: 27 in Phase 1a and 20 in Phase 1b. For each of these meetings, we tabulated the number of attendees, the number of personal computing devices (e.g., laptops and PDAs), the frequency of display and whiteboard use, and the use of non-electronic resources such as printouts and paper notebooks. We supplemented these detailed observations with five interviews with meeting room users, unstructured in-person observations of meetings, and numerous informal conversations with users.

The statistics we collected are presented in

	Phase 1a	**Phase 1b**
Number of meetings observed	**27**	**20**
Average attendees per meeting	**5.8**	**6.3**
Median attendees per meeting	**6**	**5.5**

Average # of personal devices per meeting	1.48	1.85
% of attendees with at least one personal device	26%	29%
% of meetings with at least one personal device	70%	75%

Table 4-1 andTable 4-2. As stated earlier, we were especially interested in:

- What client devices would be available to room users that would allow them to configure and control other aspects of the environment?
- What existing room resources were the most commonly used and what were they used for?
- What problems and frustrations did users encounter?
- What types of activities are carried out in meetings, and how are different technologies employed to support those activities?

	Phase 1a	Phase 1b
Number of meetings observed	27	20
Average attendees per meeting	5.8	6.3
Median attendees per meeting	6	5.5
Average # of personal devices per meeting	1.48	1.85
% of attendees with at least one personal device	26%	29%
% of meetings with at least one personal device	70%	75%

Table 4-1: Frequency of personal device usage in meetings during Phase 1 of the study. This data shows that while a minority of meeting participants had access to a personal device, at least one participant had access to one in a majority of meetings.

	Phase 1a	Phase 1b
% of meetings where projector was used	41%	45%
% of meetings where printouts were used	41%	55%
% of meetings where whiteboard was used	11%	20%
% of meetings where public PC and Plasma was used	n/a	5%

Table 4-2: Frequency of room device usage in meetings during Phase 1 of the study. In both phases printout and projector use dominated the means of data sharing support preferred by attendees. Interestingly, the addition of a publicly available PC and 50” Plasma screen in Phase 1b did not affect attendees choice of data sharing mechanisms with the exception of one meeting.

The only client devices that were seen with any regularity were laptops. One PDA was seen, and it was used for less than five minutes by someone who was also using a laptop at the same time. If cellphones were present, they were hidden and did not emerge during the meetings. Our interviews and informal conversations with participants revealed that most cellphone users at our institution leave their phones in their offices, in large part because their collaborators are mostly co-located with them at PARC and so they do not use cellphones for work-related communication during the workday. A minority (26-29%) of attendees were observed to use a personal device during meetings, though a majority of meetings (70-75%) had at least one attendee that was using a laptop. When we inquired about people's practices around carrying laptops, we found that the decision to carry or not to carry is based on a number of factors as well as personal preference. If someone was planning to present some information at a meeting in the form of a set of slides or a document, then they would certainly bring their laptop in order to do so. However, some individuals would bring a laptop if they thought there was a chance that they could use it to help the group discussion by retrieving and displaying relevant information (e.g., web sites or documents from a shared group repository). One user, "J," was well-known to play this role in a variety of groups, to the extent that other users said that they would leave their laptop behind if J was going to be there because they knew that he would be available to facilitate any information retrieval task. Still others would bring laptops if they thought that their full attention would not be required for the duration of the meeting, in order to get other things done such as responding to email or working on other projects.

We had hoped that the introduction of the public PC and plasma screen would (a) allow some users to leave their laptops behind and/or (b) increase the incidence of serendipitous retrieval and display of information relevant to a discussion. It appears to have had neither effect for most users as it was only used once over the course of observations, and this instance was most likely due to the novelty of the installation. Other users expressed reservations or even distaste at the notion of using a public PC. One user even said that, for him, a computer was "kind of like a toothbrush"—i.e., not something that you would share with other people. While this probably represents an extreme view, it was clear that among our target users the notion of a public client would not be likely to catch on in the near term. We concluded that users' laptops would be our best choice for any client software that would be used for configuring and controlling the room resources.

In addition to portable computing devices, a large portion of users used paper-based resources such as notebooks and printouts. A common pattern was that all meeting participants would show up with a printout of the same document, which had been emailed out to the group by one of the members a few minutes or hours beforehand. On some occasions, the document author would instead show up with a stack of copies of the document to hand out to other participants at the meeting. Printouts were used slightly more often than shared displays, and occasionally both modes of information sharing would be used in the same meeting. The prevalence of printouts can be attributed to a variety of factors, ranging from the simplicity and ubiquity of email as a document sharing medium to the fact that a personal printout is easy to peruse at one's own pace and annotate at will. We did not pursue the use of printed materials in meetings, though others have [31], and this continues to be an interesting area for future research.

The most commonly used room resource, by far, was the projector. Whiteboards were used on occasion and the whiteboard capture facility was used even less commonly. Clearly, printers were used in advance of the meeting, as evidenced by the prevalence of printouts. On a couple of occasions a meeting attendee would leave for a few minutes and return with fresh printouts, indicating that a remote printer had probably been accessed by someone in the meeting. Other room resources such as a speakerphone and an overhead transparency projector were not used at all.

When asked to articulate problems and frustrations with this meeting room, a common complaint was that the meeting room equipment does not "work right" or is hard to get working. Everyone could tell a story of woe about getting their or a visitor's laptop to work with the projector because of some obscure incompatibility or setting on one or both of the devices. Another frustration was the difficulty of accessing information that one had not specifically prepared for sharing at the meeting. This frustration was most acute among attendees who had neglected to bring laptops, but could happen even when a laptop was available—for example, having a document on one's desktop computer and not being able to easily access it from the room. It should be noted that neither of these problems were seen as particularly severe, and in fact some users were hard pressed to think of any problems at all. By and large, people were reasonably satisfied with the room as it was currently configured but were open to trying new things if they didn't get in the way of what was already working.

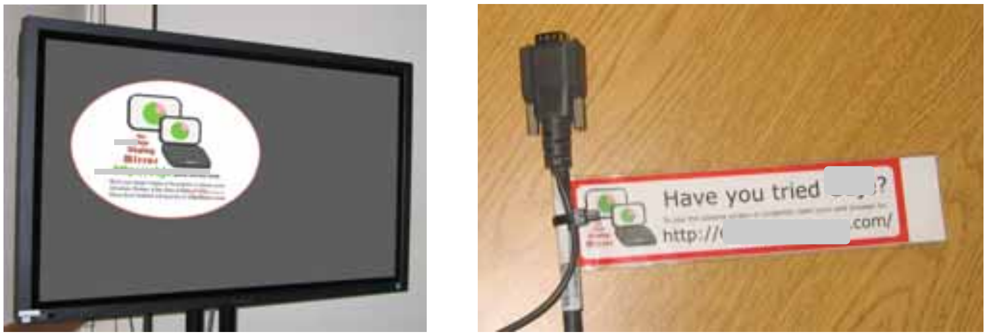
4.2.4 Preparing to Intervene

When we set out to design a set of services to deploy in and around our institution's meeting rooms, we took seriously the fact that the user population was largely satisfied with things as they were. Since our goal was not necessarily to make meetings more effective or efficient, but rather to explore how to design and deploy ubiquitous services that would be adopted and incorporated into the daily life of a work community, we were less concerned about the lack of a clear "pain point" and more concerned with ensuring that whatever technology we introduced formed a snug fit with existing practice. Therefore we made three decisions regarding the design of our service deployment:

1) Since laptops were fairly prevalent in meetings, and our users were reluctant to adopt shared computing devices, we would leverage users' existing laptops as client devices.
2) To facilitate adoption of our technology, we would begin with a small deployment of a single simple but high utility service. Since the most popular installed resource in the observed meeting room was the projector, we determined that the first service to deploy would be a service that supported the use of shared displays.
3) The fact that similar meeting activities can be accomplished with varying technologies confirmed our intuition that it would be more fruitful to provide low level tools like display mirroring or document sharing that could be incorporated into a variety of activities rather than, say, a set of high-level integrated applications that would support specific activities such as brainstorming or group editing. Thus we reaffirmed our initial goal of deploying a set of loosely-coupled, flexible services that could be composed into a variety of applications by end-users.

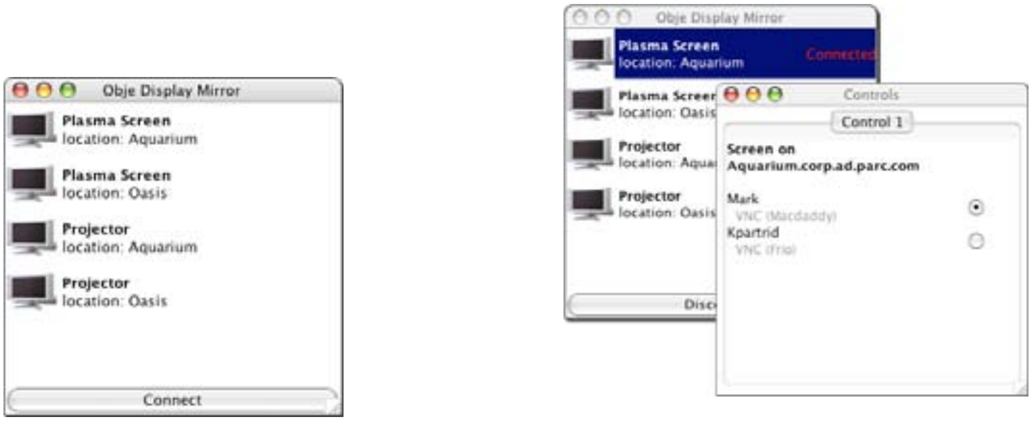
4.3 The Display Mirror

The initial service offering that we created was called the Display Mirror. This service allows any meeting participant with a networked laptop to mirror their laptop's screen to any public display that is running the service. To carry out the mirroring, the user visits a web site on our Intranet and clicks a link, which downloads and runs a client application using Java Web Start. The link address was advertised in various ways throughout the lab, as shown in Figure 4-3. In most cases, this is a very simple operation and takes less than a minute the first time it runs and can take just a few seconds each successive time. The client application is shown in Figure 4-4. Figure 4-4(a) shows the application as the user would first see it. After the user selects one of the screens and clicks "Connect," her laptop screen is mirrored to the selected screen and an additional control UI appears on her screen, as shown in Figure 4-4(b). When multiple users are connected, the control UI shows each connected user and allows the user to choose which user's screen to display on the projector. All connected users see the same control UI and are given the same ability to choose themselves or anyone else to take control of the shared display.



(a) (b)

Figure 4-3: Two forms of advertisement used to promote the Obje Display Mirror. A screen saver advertises the existence of the Display Mirror (a) as does a tag attached to the VGA cable dangling from the wall jack (b).



(a) (b)

Figure 4-4: The Display Mirror client before and after a connection is made. First, (a) the user sees a list of available screens. After connection, a control UI (b) is presented *to all connected users* of the selected screen, allowing control of the display to be shared.

In terms of implementation, the Display Mirror application consisted of two parts: the Screen component and the Display Mirror client. The Screen component was an Obje DataSink that was designed for general use with a variety of applications and in fact earlier versions of it had been used with other Obje applications such as the Handheld Browser, Casca, the Orbital Browser, etc. For the Display Mirror installations, the Screen was further subclassed into two additional components: the Plasma component, which added a screensaver advertising the Display Mirror capabilities and instructing the user about how to access the Display Mirror client, and the Projector component, which added the ability to turn the projector on and off and switch to the appropriate video input. Each Screen component that was deployed was hosted by a dedicated computer running either Windows XP or Mac OS X. The Display Mirror client was a small application that included an embedded VNC server [125], which acted as an Obje DataSource. When a user would connect his or her laptop to one of the Screen installations, the Display Mirror client would initiate a connection by starting up a local VNC server, then delivering to the selected Screen a granule that contained a Java-based VNC-viewer along with information about how to connect the viewer back to the just-started server. Upon receipt of this granule, the Screen component would instantiate the viewer, which would internally configure itself to connect back to the VNC server embedded in the initiating Display Mirror client and render the user's laptop's desktop onto the publicly shared Screen.

Viewed from a distance, the capabilities provided by this service are similar to those available through other means. There are projectors [24] and network-to-

VGA adapters [93] available for general sale that support some versions of direct screen mirroring. Well-known systems such as X11 [135], VNC [125], and Windows Remote Desktop [105] allow some or all of one computer's screen to be mirrored to another computer's screen. The novel features of the Display Mirror are that (a) the conventional direction of screen mirroring is reversed—the user "pushes" their screen to a public, shared screen rather than "pulling" a remote computer's display to their local machine; (b) multiple users can mirror their computers to a single shared display simultaneously and easily control which one of them has the ability to control the shared screen; and (c) it is a stand-alone, platform independent application with a very narrow range of functionality and minimal user cost—a genre of technology we call *micro-applications* (this concept is discussed in more detail later in this chapter). No screen mirroring capability was in widespread use among our target users at the time we were planning our deployment, and no such technology was deployed for public use in any meeting rooms.

4.3.1 Looking Into the Display Mirror

Six months after the initial deployment, the service had achieved some success in terms of adoption. It had attracted a stable core of regular users and had gained a stable position in the toolbox of meeting room technologies that are employed by users. As I will report in this section, other successes were also observed: in particular, feedback we received indicated that the experience of using the service held unexpected benefits for both the direct users and the audience with which he or she was sharing data. Long-term usage data also revealed that a subtle but significant shift in meeting room data practices had taken place as a result of the Display Mirror deployment—namely that the instances and types of multiple user screen sharing increased.

4.3.1.1 Initial Experiences with the Display Mirror

In the first few weeks of the deployment, we sat in on three meetings in which we asked one or more participants to use the Display Mirror to present any information they were planning to share. Afterwards, we asked each participant to fill out a short questionnaire and we conducted a focus group to collect initial reactions to using the technology. Since not all participants directly experienced using the Display Mirror client, we refer to those who did as the "primary" and those who did not as "auxiliary" users. All of the "primary" users felt that the process of connecting with the Display Mirror was as fast or faster than the conventional way of connecting. In addition, the experience of using the Display Mirror was felt to be more "natural" in certain ways. All said that they very much liked being able to connect wirelessly to the projector. First, the awkwardness of dealing with the physical cabling was eliminated (for instance, participants did not have to crawl under the table to retrieve the VGA cable). Second, this newfound flexibility allowed them to sit in parts of the room that had previously been unusable for them because they were too far away from the cable.

The largest subset of "auxiliary" users was essentially unaware that a different technology was being used. Others remarked that the setup seemed faster and easier. One participant remarked upon the absence of "physical thrashing about," which suggests, in this participant's words, that the new technology seemed "calmer" than the previous way of doing things.

However, we also became aware of the Display Mirror's downsides. The increased latency of the connection is an issue for certain tasks, such as group editing of a text document. The lag between when the primary users update the document and when it is reflected to the rest of the group is irritating to some users, though some find this to be unproblematic even though they are aware of the lag. For other tasks, such as showing a slide presentation and scribing notes the performance was felt to be adequate.

4.3.1.2 Sustained Usage and Experience

Two time periods after the deployment of the Display Mirror were analyzed. Phase 2a spans the four weeks immediately after initial deployment, and Phase 2b represents six weeks spanning the fifth and sixth months of the deployment.

Phase 2b represents a stable adoption state, in which novelty effects have worn off and users have had a chance to determine whether or not the service fits in with their work practices or not. The usage figures for Phase 2b shown in Table 4-3 and Table 4-4 reflect that the service was somewhat successful at integrating into users' work practices. Of the 29 display connections observed during Phase 2b, 16 (55%) were made using the ODM. This included 41% of the connections made between laptops and the projector and 100% of the connections made between laptops and the plasma screen.

To be sure, the fact that we are ourselves users of the meeting room (though our own usage data has been expunged from all statistics reported in this paper), and the fact that our colleagues are at least somewhat aware of our project's goals and that we are observing their behavior has some effect on their motivation to adopt or abandon the technology. A significant portion of the rooms' users were people with whom none of our team members have any regular contact and in several cases whom none of us had ever met. Also, our assessment is that our population of users is, by and large, eager to try new technology (thus representing an "early adopter" mentality) but is also quite picky about what they actually adopt on a long-term basis (cf. the earlier quote about a computing environment being "like a toothbrush"). Thus we feel confident that any users who were continuing to use the Display Mirror after six months had decided that it was a good fit for their practices, and in fact conversations with several of the core users confirmed this belief.

	Phase 2a	Phase 2b
Number of meetings observed	14	33
Average attendees per meeting	5.9	6.67
Median attendees per meeting	6.5	6
Average # of personal devices per meeting	2.00	2.67
% of attendees with at least one personal device	34%	40%
% of meetings with at least one personal device	71%	82%
% of meetings where projector was used	43%	42%
% of meetings where plasma was used	14%	12%
% of meetings where any display was used	57%	42%
% of meetings where the Obje Display Mirror was used	7%	24%

Table 4-3: Personal device and display usage statistics for Phase 2a and 2b. Phase 2a spanned the four weeks immediately after the deployment of the Display Mirror. Phase 2b represents six weeks of observations after the service had been available for six months.

	Phase 2a	Phase 2b
Total # of projector connections made	8	2
Total # of plasma connections made	2	7
% of projector connections carried out using ODM	13%	41%
% of plasma connections carried out using ODM	0%	100%
% of all display connections carried out using ODM	10%	55%

Table 4-4: Adoption figures for the Display Mirror (ODM). In Phase 2b, the ODM was used for 55% of all display connections. Note that in contrast to Table 4-3 this table references the total number of *display connection events* rather than the number of meetings in which displays were used.

Table 4-3 and Table 4-4 tell the story of the Display Mirror's effect on practice. In Phase 2a, almost no usage was recorded. This is interesting because it appears to demonstrate an anti-novelty effect—rather a reluctance to try an unproven technology. Therefore we used the new service on a regular basis in order to gradually introduce other meeting attendees to its capabilities and demonstrating its use. Eventually it appears to have caught on: by Phase 2b, the Display Mirror is used in over half of the meetings in which shared displays are used, and in nearly a quarter of the meetings overall. Notably, the number of meetings in which one or more shared displays are used has not changed much from Phases 1a and 1b—a shared display is used in 40-50% of meetings regardless of phase—, it is simply the case that the Display Mirror has replaced the VGA cable as the means of connecting to shared displays in a portion of the meetings.

The most interesting result of our deployment, however, is depicted in Table 4-5. We observed an increase in the number and type of multiple user display events in Phase 2b when compared to all previous phases. We define a "multiple user display event" as occurring whenever more than one individual connects his or her personal device to a shared display during the course of a single meeting. Examples of such events include serial display events (Alice shows her slides on the projector, then Bob takes over and does the same), overlapping events (while Alice is showing her slides on the projector, Bob shows a web page on the plasma screen), and interleaved events (Alice shows something on a display, then Bob takes over the same display, then Alice resumes control). Before the introduction of the Display Mirror, the only type of events that were observed were serial display events, and they were observed in approximately 10% of meetings. After the introduction of the Display Mirror, all three types of multiple display events were observed, and at least one type of multiple display event was observed in 24% of all meetings. Overlapping events require at least two screens, so they could not have been observed in Phase 1a, but could have been observed after that. In fact, they were not observed at all until Phase 2b, when they appeared in 4 meetings (12% of all meetings). Interleaved events could have been observed at any time, but the awkwardness of unplugging and replugging VGA cables between laptops could account for the fact that such events were not in fact observed until Phase 2b, when they appeared in 2 meetings (6% of all meetings).

Phase	*1a*	*1b*	*2a*	*2b*
Total # of meetings	27	20	14	33
% of meetings with *any* display use (either projector or plasma)	41%	50%	57%	42%
% of meetings with *multiple display events* (includes serial, interleaved and overlap events)	11%	10%	7%	24%
% of meetings with *serial* display events	11%	10%	14%	6%
% of meetings with *interleaved* display events	0%	0%	0%	6%
% of meetings with *overlapped* display events	n/a	0%	0%	12%

Table 4-5: Types of multiple display events with and without the Display Mirror. Multiple-user display events (including serial, interleaved, and overlapped events) increased overall, and two new types of multiple user display events were observed.

While we did not seek to explicitly characterize the situations in which the new display capabilities were used, it became clear through informal observations and conversations with users that people were appreciative of the new practices that had been made possible. For example, one pattern that was observed after the Display Mirror deployment was the quick display by a meeting participant of a web page to the rest of the participants that was relevant to the current discussion. This type of thing was possible before the Display Mirror but was almost never observed until after it was available. Display Mirror users who experienced this new capability reported that it was a useful aid to meeting discussions, and that the low overhead involved in displaying information to other meeting participants was the key to enabling the new practice. Another new practice that emerged after the Display Mirror deployment was for meeting participants to display different types of information on the two displays, for example displaying an agenda on the plasma display while displaying a series of slide presentations on the projector. This practice was also possible before the Display Mirror was deployed (though only after the second display was available) but was not observed until after the Display Mirror deployment.

We view the emergence of these new use patterns as the most promising outcome of our deployment thus far, as we believe it underscores the potential of small, subtle changes in mundane aspects of interactions with technology to effect subtle but significant impacts in the execution of higher-level human tasks and activities.

4.3.1.3 Challenges Faced

By and large, we were able to achieve our goals with the Display Mirror. We introduced a technological intervention in the form of a lightweight, continuously available networked service, and this service was adopted by our target users, incorporated by them into everyday practice, and used to improve and extend existing practices in new ways. However, these gains were achieved at some cost.

The effort required to collect, manage, code, and analyze a year's worth of data, even given our efforts to streamline the amount of data that was collected and the analysis process, was greater than expected. Much of this effort was expended in automating the data collection, migration, and backups, as well as in building and improving the analysis tool shown in Figure 1. The coding itself was not overwhelming—a one-hour meeting would take approximately 30 minutes to code. We coded in pairs so that we could more easily notice and discuss patterns and interpretations of the data. Since a typical week would contain 5-10 meetings, the coding amounted to 6-12 person-hours per week of data (3-6 pair-hours, which includes time for loading data into the tool and searching for meeting start and end times). This is considerably less than reported by Consolvo, et al. in their use of LSA [5], though our data requirements differ in that we were fairly familiar with the activities being studied and we were able to articulate in advance what types of events we were looking for.

The effort required to produce and maintain a deployment was also quite substantial. Even a fairly modest deployment involving 10-15 client devices and 4 display mirror services in two meeting rooms required approximately 50% of one engineer's time and 10-25% of another's over the course of the six months of deployment reported in this chapter. Part of this was due to the fact that even a small deployment encounters many of the problems that a larger one would encounter: client platform compatibility problems, problems with the corporate network configuration, and issues with the stability of the service nodes (which were running Windows XP) to name a few.

In essence, the high degree of effort to create and maintain the data collection infrastructure as well as the deployment itself, represent poorly amortized costs. We anticipate that these costs would not be incurred to such a large degree for the deployment of the 2nd, 3rd and nth service. However, for a single service with such limited utility, we regard the costs as having been unreasonably high. The observation that longitudinal studies and robust, sustained, networked deployments are expensive should not be news to anybody. What is worth noting, however, is that such techniques appear to be essential for the design and evaluation of ubicomp technologies. This indicates that the search for improved data collection and analysis techniques and better deployment platforms continues to be critical to the success of ubicomp research.

Another challenge that we faced appears to be more fundamental to the goal of our research. It was somewhat difficult to achieve and sustain adoption of the Display Mirror over a long period of time. We do not believe that this was as much due to limitations of the technology as it was due to its invisibility. Connecting one's laptop to a shared display, while a common activity in aggregate, is but a sporadic, occasional activity for any one individual. It was rare for anyone in our study to connect to the projector more than once per week. Thus it is easy to forget that a service such as the Display Mirror even exists between opportunities to use it. It is more likely that a user will default to the ingrained habit of reaching for the VGA cable than that she will consider the various options for connecting and select the one that provides the greatest utility. The very nature of mundane tasks is such that they are not foregrounded in the user's conscious mind, thus making it difficult to replace the old habit with a new one. We found it necessary to temporarily increase the visibility of the display connection activity through advertisements in the meeting room, and public and personal reminders in order to gain any traction whatsoever with our intended users.

4.4 Discussion

I will conclude this chapter by dwelling on a few key aspects of the experiences with the Obje Display Mirror that are worthy of further discussion. These aspects include implications for

- the design and evaluation of "mundane" ubicomp technology,
- the challenges of delivering and sustaining a robust, long-lived installation of ubicomp technology,
- the ways that low-level capabilities like display mirroring are packaged and presented to users of ubicomp environments,
- and the value that was provided by Obje and the broader notions of "Recombinant Computing" to the Display Mirror project.

4.4.1 Studying Boring Things

In her article on the ethnography of infrastructure, Star called on researchers to "study boring things" [139]. She proposed that computers are frequently less of an "information highway" and closer to "symbolic sewers." As such, she argues that we need to pay more attention to "the plugs, settings, sizes, and other profoundly mundane aspects of cyberspace."

Our experience in deploying and evaluating the Display Mirror resonates with many of Star's comments. A service that allows users to bypass the VGA cable when connecting to shared displays is probably not ubicomp's "killer app". The most widespread activities we observed in meeting rooms (e.g., connecting a laptop to a screen using a VGA cable) are indeed profoundly mundane; the patterns of interconnection between devices we coded could easily be categorized as boring. Yet it is exactly these mundane activities, prevalent yet ignored, that many ubiquitous computing systems could be best suited to support. This, in turn, requires alternative methodological and design approaches.

1) ***Longitudinal observations highlight background activities.*** Mundane practices pose several methodological difficulties. As they are diffuse and often pushed to the background, they require long and repeated observation to be uncovered. Traditional interview techniques and laboratory studies, for instance, cannot foreground these activities that, for the most part, are not attended to by their participants. Instead, longitudinal observations are required. But these cannot be entirely automated, as only careful qualitative analysis of the data will progressively reveal these widespread tasks unconsciously carried out by the users. While we have described an approach to mitigate the problem by blending automatic data collection, interviews, and qualitative data coding, it remains that longitudinal studies are inherently costly in terms of manpower. The observe/design/evaluate cycle that is foundational to HCI research might

be significantly longer when designing for the support of mundane activities.

2) ***Alternative means of assessing "improvement" are needed.*** By definition, mundane practices have reached a point where they are simple or integrated enough that they disappear from consciousness. While they can sometimes come to the foreground in the case of novice users or outliers [95], the main user population simply forgets about their cost and implications. As such, it is extremely difficult to justify alternative ways of carrying out mundane tasks using traditional success metrics. The time required to complete a task, for instance, may very well not be significantly lower in ubicomp-supported scenarios such as ours than in the standard case, since the user cost for the latter is already extremely low. However other, less obvious, benefits may accrue to the users and other members of the users' environment. For instance, *the beneficiaries of these ubicomp systems might not be their direct users*. In the case of the Display Mirror, displaying information on a projector is not only useful to the laptop's user: the other meeting participants are also affected. As we learned, the audience was able to articulate benefits of the Display Mirror that we had not anticipated: now that public displays' users were unwired, meeting participants described a less "chaotic", more "fluid" meeting experience

[161]. Furthermore, *benefits to the direct users may go beyond human-computer interaction*. Users of the Display Mirror identified increased mobility within the room as a benefit. By being untethered and therefore able to use the space in the meeting rooms more flexibly (e.g., not to be forced to sit in the "presenter's spot"), their experience of the meeting was improved. Space is a very important social resource and symbol, used to signify status and roles [60]. "Untethered" computing gives control of the space back to the users. While this has nothing to do directly with human-computer interaction per se, it is certainly a benefit. Ubicomp researchers need to consider the global benefits of their system, beyond the confines of a single user interacting with a machine, or even a whole network of machines.

3) ***Sustained adoption of mundane technology is especially challenging.*** It has often been proposed that a mark of success for ubicomp systems is when they "blend in" their environment. However, when activities are already "blended in," it creates a significant challenge with regards to driving adoption of a new technology.

While we initially tried as best we could not to be disruptive when deploying our infrastructure, it quickly became clear that this was not likely to bear fruit in a reasonable time frame. Mundane activities are deeply entrenched. If new technology to support them remains invisible, it simply won't be adopted.

Therefore, to drive the adoption of our system, “infrastructural inversion” [14] was necessary: we had to temporarily foreground the backstage elements of our users’ work practices, for instance by attending meetings ourselves and repeatedly pointing to our use of the new technology. Without such insistence and somewhat “heavy handed” behavior, nobody would have known that a new technology was in use—after all, the end result was no different from earlier meetings (some information appeared on a public display). While this shows that the Display Mirror was truly transparent, it obviously did not favor its ultimate adoption.

Therefore, unlike naturalistic ethnographies of systems’ deployments, ubicomp researchers may need to be forceful and directly intervene in order to show people the new possibilities when dealing with mundane activities. Systems such as the Display Mirror deal with the “taken for granted” part of computing systems, and human practices in this area are extremely inertial. This is unlike entirely new devices and/or applications that support new (foreground) tasks, whose novelty or strangeness makes them inherently visible. A paradox of mundane ubicomp systems design and evaluation, in our experience, is that one needs to be simultaneously forceful and gentle, that is, to highlight new ways (intervention) of interacting that will end up being as mundane as what they are meant to replace.

An application or system designed to support mundane tasks may face additional challenges when compared with systems that are designed to be used in a focused way to accomplish a set of conscious tasks. This is especially true when the application is a replacement for an existing system that is still available for use. In the case of the Obje Display Mirror, the demands for ease-of-use and robustness were especially high, since the users could easily abandon our system in favor of the previous, familiar, and still available VGA cable. If any difficulties were encountered during use, users would simply abandon the Display Mirror and revert to earlier practices. Once this reversion had taken place, they were reluctant to re-try the Display Mirror without another explicit intervention. In systems such as the Display Mirror, the tasks being supported are, by their nature, in the background and not the focus of conscious attention, and as a result users are extremely intolerant of any glitch that forces them to pay attention to an otherwise unconscious activity.

4.4.2 Challenges for Sustained Deployment, Usage, and Evaluation of Ubicomp Technology

Some of the lessons we learned in this project would apply equally well to other types of information technology, but some appear to be if not unique to, at least more pronounced in, ubiquitous computing environments.

Longitudinal studies of users' practices before and after the adoption of a new technology are almost always considered desirable, but generally viewed to be expensive and difficult. A common practice in user-centered design is to conduct small, iterative evaluations with progressively refined prototypes. This allows designers to identify critical shortcomings in functionality that will prevent users from accomplishing tasks that the system is being designed to support. However, given the background nature of the tasks we sought to support in the Obje Display Mirror, which we believe are of a similar type to tasks supported by other ubicomp systems, such design and evaluation methods are not likely to yield useful results. Much more so than in conventional desktop or client-server applications, ubicomp applications are explicitly designed to be absorbed into everyday practice, and so the design and evaluation methods need to be more closely interwoven into the background of the practices and environments being supported. In other words, longitudinal methods that are desirable in desktop systems become essential in ubicomp.

Following on the above point, the inadequacy of standard usability metrics such as task performance time and error rate for characterizing the acceptability of a software system has been acknowledged in both the HCI community (e.g., [8]) and ubicomp community (e.g., [2, 25]). However, this point is worth making again, as in cases where the tasks being supported are not the explicit focus of attention, traditional usability metrics and methods are even less adequate as they provide little information about how the system being designed will affect the larger context in which it will be used. Another way to say this is that in ubicomp, the larger context of use may be even more important than in other domains, and so the design and evaluation methods used must be selected or modified to take larger context into account.

The difficulty of getting users to adopt a new technology, especially when a viable and established alternative exists, is well known in multiple domains (e.g., [56]). The issues in ubicomp are probably not much different. One slight but important difference, however, comes back to the issue that the tasks being supported may not be conscious or explicit. In this case, the necessity to foreground the task in order to force the user to become aware of a new technology, and to choose between the existing and new methods of accomplishing the task, may cause an unwelcome interruption in the flow of some other activity. This implies that the methods used to foster adoption need to be as unobtrusive as possible and need to fit in well with existing practice. In our project, we attempted to do this by attaching advertisements to the devices whose functionality we were attempting to replace or enhance. Even this proved to be too subtle, however, and we had to use more forceful methods—namely social pressure. On the other hand, the fact that our system was designed for public use was helpful from the perspective of disseminating information about its availability, usefulness, and usage instructions. Another approach that has been taken to increasing adoption of ubicomp technologies is reported in [96], in which the authors introduced a number of different applications aimed at increasing the adoption of an enabling technology—in this case, wireless location tracking badges.

However, we did not wish to introduce a number of different applications that used the Display Mirror, because this likely would have caused Display Mirror usage to become more of an explicit goal rather than a means to an existing work-oriented goal.

Creating and maintaining robust, available, easy-to-use services is always challenging, but certain factors in ubicomp may make this even more challenging. For one thing, the expectation of reliability may be higher, since in at least some cases the practices being replaced are ones being carried out with more reliable tools (e.g., analog or physical). For another thing, ubicomp systems often stress standard assumptions about device, system, and application configurations. They may, for example, rely on the coordination of multiple distributed processes, and may depend on assumptions about network topologies and security policies that are sufficiently different from existing application models (e.g., client-server or peer-to-peer) that standard configurations will not support them. In our case, we had two such considerations that interfered with the functioning of our application—one was the fact that we relied on multicast (mDNS) for the Display Mirror clients to discover available display services and the other was that we depended on being able to open sockets on otherwise unused ports to stream the video data from the laptop to the display service. In the case of multicast, we discovered that our application employed a pattern that had not been encountered by PARC's Networking Support department—namely the ability to pass multicast traffic back and forth between our wired and wireless networks—and we had to raise the issue through several levels of management to get the wireless bridges configured properly to support our application. Regarding the need to open unused ports, it is PARC's policy to install and configure a personal firewall on all laptops. Some of these firewalls silently block traffic on all ports unless the user explicitly enables it using an advanced control panel. Other firewalls would allow traffic on these ports but only after the user had agreed to a series of rather cryptic dialog boxes. In the common case where the firewall was configured incorrectly to allow our application to run, the application would fail silently and the user would assume that our service was down.

There are certainly other classes of applications where mysterious and seemingly unrelated system settings can impede proper functioning of the application, but it seems that for the time being ubicomp applications may run up against these problems more frequently than most—at least until the coordination patterns and configurations required to support them become commonly understood.

4.4.3 Supporting Micro-tasks: Micro-applications or General Tools?

We believe that our experiences can inform a range of existing and forthcoming ubicomp applications that focus on support for infrequent, mundane tasks, especially those that form constituent parts of a variety of foreground tasks. In our observations, for example, connecting a laptop to a projector was a subtask of a variety of larger activities, such as “giving a presentation,” “scribing notes,” and “sharing information” (e.g., by showing a web page to a group).

It has been previously noted that task-oriented application design and evaluation may not be appropriate for many ubicomp scenarios, because ubiquitous computing is at its best when it fades into the routines of life—that is, when it supports ongoing activities through continuously available, yet sporadically accessed services [2]. Many of the mundane activities we observed do not fit well into the standard HCI notion of a “task” (as characterized, for example, by Lewis and Reiman [87]). Still, they are activities that users perform, and perform quite often, and which—despite the fact that they are not foreground activities—still to some degree determine the experience that users have in a space. It is perhaps more appropriate to regard these activities as *micro-tasks*, because they’re rarely explicitly attended to by those who perform them, are generally short-lived, and are typically merely a step in the process of accomplishing some larger tasks (such as giving a practice talk for colleagues). With the Display Mirror, we have taken a look at the micro-task of appropriating a public display to share information with others in a meeting room, and we have designed a single, small, limited-functionality service to improve the experience of carrying out that micro-task. Correspondingly, we propose that a service like the Display Mirror should be thought of as a *micro-application*.

While the term “application” can take on many meanings at many granularities, it has frequently been used by technologists and researchers to mean a substantial, user-visible collection of functionality, along with some user interface, designed to allow the user to accomplish some set of related tasks during a focused interaction. Such a meaning does not seem to be applicable to a tool designed to support a single, simple function that is not a part of a consciously attended action on the part of the user. Shifting from applications to micro-applications, particularly when we imagine environments in which myriad micro-applications co-exist, converge, and even compete for the users’ attention, causes us to rethink much of what we know about application design, deployment, and evaluation.

While the “micro-application” terminology may be new, similar conceptualizations exist throughout the research literature. Abowd and Mynatt [1], for instance, describe the “informal and unstructured activities typical of much of our everyday lives,” and distinguish these from the dialog styles that are normally a focus of HCI research. Others [39] have argued that small, lightweight applications are an appropriate unit of analysis in ubicomp settings. From purely an evaluative standpoint, separating functionality into such micro-apps may make it easier to tease apart the effects of the application and the role of the infrastructure, by limiting confounding factors.

Our experiences with the Obje Display Mirror provide some insight into the methods and approaches that will be effective for the design and evaluation of micro-applications in ubicomp environments, but I believe that there is a good deal of further investigation that can be done in this area.

A larger question, however, is how best to provide micro-applications to users. In the Display Mirror example, a discrete client application was provided explicitly for the task of connecting one's laptop to a shared projector. One can imagine other ways that such functionality could be delivered to users. It could be integrated into a standard operating system such as Windows XP in the form of a new control panel or even an extension of the existing Display control panel. It could be provided as part of a larger application that allow a user to perform various tasks related to workplace collaboration or data sharing (e.g., Casca or the Sharing Palette). Alternatively, this functionality could be provided as but one of the capabilities of a general service discovery and composition tool that allows users to access various networked services and compose them in ways that are appropriate to the task of any given moment. While this latter approach will give users the greatest control and flexibility, it runs the risk of failing to communicate effectively with users about what capabilities are available in the environment and to provide them with a clear means of effecting any particular "micro-application" that they may need to access at any particular time.

Resolving this tension between providing users with the access to the maximum flexibility and power available in an environment of networked devices and services and providing clear and efficient access to any given capability at any particular time is the primary objective of the OSCAR project described in the next two chapters. Though we will be shifting domains from the workplace to the home, I believe that the issues addressed and the techniques proposed in OSCAR are highly applicable to these questions about how to best deliver a range of functionality that have been raised in the context of the Display Mirror project.

4.4.4 The Obje Advantage

As mentioned previously, the Display Mirror is a specialized application built atop Obje that does not expose notions of service composition to end-users in any significant way. While, technically speaking, a service composition is being effected each time a connection is made between a Display Mirror client and a Screen component, this fact is probably not clear to most users. Rather, the user's experience of the Display Mirror is one of discovering a set of available screens, selecting one for use, and invoking a single, fixed operation: "mirror to screen."

Nevertheless, building the Display Mirror atop Obje provided several advantages, both from a technical and a user experience standpoint. From a technical standpoint, the prior existence of the Screen component, and the ability to incorporate it with very few modifications into the Display Mirror application was a great success in terms of code re-use. The Screen did not have to be rewritten to understand the VNC protocol, data format, or anything about VNC at all. This was all handled by the implementation of the Display Mirror client. Meanwhile, the unmodified Screen was still available for other uses, and in fact it was again re-used with little modification for the OSCAR project described in the next two chapters. Similarly, the aforementioned work that had gone into re-implementing Obje for the purpose of making it easier to deploy and maintain installations of Obje services and clients made it possible to carry out the Display Mirror project. Even though we experienced significant challenges with regards to deployment and maintenance anyway, I sincerely doubt it would have even been possible given the resources available to the project to develop the Display Mirror to the point of being able to support users, let alone support them for half a year, without the leverage provided by the previous work on the Obje Framework.

Finally, derived in large part from the technical advantages listed above, the user experience of using the Display Mirror greatly benefited by the existence of Obje. The fact that new Screen installations could be easily added or subtracted from the environment without affecting clients, and that new clients could similarly be added or subtracted without affecting the installations, resulted in a user experience of seamless discovery and use of available networked resources. This experience is still somewhat rare in commercial and commonly used systems, and is precisely the kind of experience that Obje and similarly styled ubicomp frameworks are designed to provide. As we shall see in the next chapters, however, I believe that this experience can be improved even further by also providing users with the ability to compose and control discovered resources in flexible, customized ways.

4.4.5 Contributions and Limitations

The Obje Display Mirror project focused on designing and evaluating an application that delivers an integrated user experience of interacting with distributed resources. Moreover, the project was concerned with redesigning the experience of an existing practice: sharing the display of one's laptop with collaborators in a meeting room. The original goal was to examine existing practices among the members of PARC's Computer Science Lab and introduce new technology in order to improve and facilitate those members' interactions with embedded technology.

In order to understand existing practice as well as to evaluate the effects of introducing new technology, a novel observation method was created that allowed meeting room device usage to be monitored across several months. The experiences with longitudinal video sampling in the development of the ODM are important because they highlight the difficulty of monitoring sparse, distributed behavior over a long period of time. Yet understanding such behaviors is critical to the enterprise of understanding, developing, and evaluating an "integrated" user experience of interacting with multiple devices to carry out everyday tasks.

The Obje Display Mirror was an appropriate case study for examining the effects of providing an integrated user experience. The ODM replaced a disruptive mechanism for connecting one's laptop (connecting to a VGA cable) with a mechanism (manipulating a software client) that was better integrated into users' simultaneous activities (locating and preparing data to share). The results of the study of ODM adoption and use showed that redesigning the display connection experience provided benefits for both the primary users of ODM (individuals wishing to display information to others) and for their collaborators (the individuals to whom information was to be displayed). It also showed that substantive changes in practice could be achieved over a period of months due to the provision of subtle but important changes in the functional capabilities of the service being provided—namely in this case, the ability to support multiple simultaneous connections with distributed control over the connection being displayed on the screen.

The limitations of the study were also instructive for future researchers. The end result of the study—a service that allows users to connect wirelessly to projectors and other public displays—was perceived by users as a marginal improvement over the system that it had replaced. This is due in large part to the fact that the previous system for accomplishing the same task was not particularly cumbersome or difficult for most users who had already learned how to carry it out. Given the low perceived incremental value of the service that was deployed, the effort required for design and evaluation was somewhat labor-intensive, as it required human review of each meeting to code for device usage events. While it was considerably less intensive than similar methods that had been used for similar studies (e.g., LabScape [6]), the perceived value of the end result would probably not justify the investment for widespread use. In addition, the amount of effort required to deploy and maintain the ODM service and, especially, the multi-platform client, was substantial. The primary instructive lesson from these experiences is that designing and evaluating distributed technologies for extended deployment and use is challenging and expensive, and that designers should take care to ensure that the value of the deployed technology to end-users outweighs the costs.

4.5 Summary

The experiences with the Obje Display Mirror reported in this chapter demonstrate that a focused service deployment aimed at supporting mundane microtasks such as connecting one's laptop to a shared display can have concrete, positive impacts on users' work practices and on the user experience of interacting with their environments. The introduction of the Display Mirror impacted the frequency and types of multiple-user access to shared displays, and affected the experience of integrating display use into meetings for both the “primary” and “auxiliary” users of the displays.

In terms of the grander themes of this dissertation, however, the ODM experience does not tell us very much about the feasibility of end-user composition or the benefits of having integrated composition and control of multiple heterogeneous devices and services on a single network. I will take on these challenges in the next two chapters take on these challenges, as I describe the design, implementation, and user study of OSCAR.

5 Designing and Building OSCAR

While specialized applications like the Obje Display Mirror can provide significant value to end-users, they suffer from the key limitation that they can only do what they were programmed to do. As argued in earlier chapters, in a world with ever-expanding devices, whose combinations entail ever-increasing capabilities, the approach of delivering specialized applications to satisfy every user need under every circumstance will be unmanageable. Application development is slow and expensive and leads to a fragmentation of user experience (the problem of piecemeal interaction discussed in Chapter 1) relative to an approach that gives users the tools to compose and control their own applications from re-usable components. The primary problem with giving users control over application composition is that the complexity and additional work of composition may limit such approaches' acceptability for large numbers of intended users.

In this chapter, I will describe the motivation for, and design and implementation of, OSCAR—a system that allows end-users to compose devices and media in the home[6].

5.1 Motivation for OSCAR

The primary motivation for OSCAR was to build a tool that would allow me to explore tradeoffs between flexibility and complexity in an end-user composition system designed for non-technical users. OSCAR was also designed to satisfy two additional goals: to provide insight into the user experience goals of Obje as a whole, and to serve as an effective tool for home media consumption and control.

5.1.1 Stuck in the Middle

As I and my co-authors discussed in [39], it is a significant challenge to evaluate the user experience goals of infrastructure technologies. And yet, it is not infrequently the case that such technologies are motivated by the desire to provide or enable particular styles of user experience. The two examples discussed in [39], the Context Toolkit [134] and Placeless Documents [36], were primarily intended to provide infrastructure support for compelling user experiences of one kind or another. In the case of the Context Toolkit, the goal was to provide

[6] The work in this chapter and Chapter 6 was primarily conducted by me, with some assistance from Trevor Smith and Ame Elliott. This work has not been previously published.

support for applications that would react seamlessly to users' situations (location, time, identity, etc.), and in the case of Placeless Documents it was to fundamentally change users' relationship with document storage from a file/directory based scheme to a scheme based on document properties.

Both of these systems found it challenging to evaluate their effectiveness in terms of their core objectives to the extent that those objectives reflect user experience concerns. Traditional metrics used for infrastructure evaluation such as system performance or work reduction for application development (as measured for example, by comparing the lines of code required to create particular applications with and without the infrastructure), while perhaps important in a wider sense, do not address the key goals or claims of these systems, or of similar systems such as Obje. Rather, it is important to find ways to evaluate the user experience claims of infrastructures that make such claims. In the aforementioned paper, we provide a set of lessons regarding how to evaluate such claims.

- Lesson 1—Prioritize core infrastructure features.
- Lesson 2—First, build prototypes that express the core objectives of the infrastructure.
- Lesson 3—Any test-application built to demonstrate the infrastructure must also satisfy the usual criteria of usability and usefulness.
- Lesson 4—Initial proof-of-concept applications should be lightweight.

- Lesson 5—Be clear about what your test-application prototypes will tell you about your infrastructure.
- Lesson 6—Do not confuse the design and testing of experimental infrastructure with the provision (i.e., creation, delivery, support, etc.) of an infrastructure for experimental application developers.
- Lesson 7—Be sure to define a limited scope for test applications and permissible uses of the infrastructure.
- Lesson 8—There is no point in faking components and data if you want to test for user experience benefits.

Throughout the development of the Obje Framework, efforts were made to take account of these lessons. Several of these lessons (1, 4, 6, and 7) are about maintaining focus on the goals of the infrastructure, and not getting distracted by secondary or tangential features. One of them (8) is about the fidelity of data and prototypes that should be used to assess user experience issues[7]. Most importantly to the subject of this chapter, several (2, 3, 4, and 5) are about how to use prototypes to gain insight into the infrastructure's user experience goals.

[7] In hindsight, I no longer agree with the wording of Lesson #8, as I believe that judiciously faking data and components is often necessary to facilitate iterative development, especially during early phases of development. Perhaps Lesson #8 should be reworded to state something like: "Simulated data and components should be used with care, and efforts to use the most realistic elements possible at all phases of development should be pursued."

In the latter part of Chapter 3, I described a number of lightweight proof-of-concept applications that were developed as part of the iterative process of developing the Obje Framework. Prototypes such as Casca, the Settop Box, and Nexus/Wander were developed with the primary goal of giving the research team insight into how the different parts of the Obje Framework would be put together for different types of applications. In each case, improvements to the Obje architecture and API were made as a result of the prototyping effort. Also in each case, a conscious effort was made not to expend the effort to develop these applications to the point where they would be robust enough to support users for any extended duration. Development was intentionally constrained to provide insight only into the infrastructure issues that were of pressing concern at the time and to demonstrate key concepts of Obje to others, and to go no further.

In other cases, such as the Obje Display Mirror and the subject of this chapter, OSCAR, a conscious decision was made to develop user-focused prototype applications with the goal of evaluating one or more of Obje's user experience claims. Thus, in the design and development of OSCAR, I was consciously attempting to apply Lessons 2 (build prototypes that express—i.e., whose objectives match—the infrastructure's core objectives) and 5 (be clear about what your test applications will tell you about your infrastructure). In addition, as I shall discuss shortly, I applied Lesson 3 (any test application must also be usable and useful).

Specifically, the user experience goals (or "core objectives") of Obje that OSCAR is designed to examine include:

User expectation of interoperability: Obje endeavors, ultimately, to set users' expectation when encountering new, unfamiliar devices, or when adding them to a network with other devices, that these devices will work seamlessly with devices the user is already using [42].

Integrated connection and control: Client devices or applications can discover and connect compatible services on the network. Clients are then able to provide an integrated experience of interacting with multiple devices by allowing the user to monitor and control multiple ongoing connections as well as receive and display user interfaces for all of the services involved [111].

Flexible, intuitive composition of devices, services, and content: Thanks to the reduced requirements for up front agreements among interoperating services, Obje is able to present a highly simplified view of networked services based on the roles they are able to play in data transfer connections. This should allow users great flexibility in the connections they can make, while also presenting an easy-to-understand model of what things can be composed together in what ways [112].

In keeping with Lesson 2 (prototypes should express core objectives), therefore, the goals of OSCAR are essentially identical to the goals of Obje as a whole. That is to say, that OSCAR was designed to provide an experience of seamless interoperability, integrated connection and control, and flexible, intuitive composition. However, while Obje is focused broadly on providing these features in a variety of domains, OSCAR has focused more particularly on the domestic environment. In the next subsection, I focus on the aspects of current and near-future home networking environments and discuss why the home domain is a particularly fruitful domain for exploring issues of interoperability, integrated control, and end-user composition.

5.1.2 Focusing on Home Networking

Home media networking is an ideal domain for examining issues such as delivering an integrated user experience of interacting with multiple devices and providing support for end-user composition.

In-home media consumption practices are undergoing a massive change driven by a confluence of factors. On the one hand, data networks in the home are becoming more powerful and are growing to include not only traditional computing devices like desktop and laptop computers, but also media-oriented consumer electronics devices such as personal video recorders [147], MP3 players [10], and speakers [46]. At the same time, media itself is being transformed into a digital commodity that can be accessed on demand in a variety of ways from a variety of sources, both inside and outside the home network.

Increased ease of access is enabling new forms of media consumption and sharing, but these new capabilities come at the expense of increased complexity in creating and managing the connections among these devices and media services. As we move from dealing with our media networks as nests of wires and piles of remote controls to dealing with them as abstract services on a generic network, we may be trading the physical entanglement of contending with wires and plugs for the logical entanglement of dealing with numerous connection options and huge collections of easily accessible digital media.

5.1.2.1 Requirements Derived from Studies of Home Technology Use

Some of the key design criteria for OSCAR were derived from previous studies of home technology use. Specifically, OSCAR is designed to support *flexible and generic composition and control*, and *reusable compositions to encapsulate common activities*.

Flexible and generic composition and control: Rode, et al. [129] reported that collections of devices vary significantly from home to home, a finding that was echoed by a study published by myself and my co-authors [55]. In addition, the latter study shows that householders' goals and desires with respect to their networks span a wide range. This requirement echoes Obje's user experience goal of providing flexible and intuitive composition of devices.

Reusable compositions to encapsulate common activities: Often the responsibilities for setting up, configuring, and maintaining home networks and devices are distributed among different household members [55, 128]. The primary reasons that users program domestic appliances are to reduce configuration time later and to automate repetitive tasks [129]. Also, it has been reported that users tend to describe activities in terms of functions rather than devices [150]. Reusable configurations are a promising mechanism for supporting the temporal and social divisions of labor that are important to households. Supporting users in constructing activities with meaningful labels should allow them to interact without having to think explicitly about devices. This requirement represents an additional requirement for OSCAR, above and beyond the Obje goals that were discussed previously.

5.1.2.2 Form Factor

OSCAR was designed to be used as a portable, hand-operated, "remote control" device. The goal behind this design decision was to reinforce the association with the notion of remote control for OSCAR's users, while expanding this notion to include the additional concepts of automatic discovery, ad-hoc connections, and re-usable compositions.

There are two approaches to managing media devices and experiences that are converging in home media networking. From the software industry, we see media-specific applications (e.g., iTunes, iPhoto [5], and Windows Media Player [104]) that allow users to manage and experience a particular type of media but have limited mechanisms for connecting with various devices in the home. From the consumer electronics industry, remote controls (including both device-specific remotes and universal remotes such as the Harmony [18]) allow users to manipulate device functions from a distance, but are not well suited for managing and redirecting media streams, especially from sources like digital media libraries.

In a sense, OSCAR seeks to capture the primary advantage of remote controls: ability to access and control your media and devices from a distance while also capturing the new capabilities and requirements of media networks—the ability to control the interactions between devices and services, not just a single device. As I write these words in 2007, my target for this work is the "early majority" home of 2010-2012. Thus I assume a fairly high penetration of home networking technology and connected media devices, but do not assume the existence of facilities such as sensing frameworks or context-awareness that I feel are somewhat farther away from market readiness. As a consequence, I assume very little automatic behavior on the part of devices in the network, and instead assume that user interaction will invoke operations and control devices.

I elected to design for a touchscreen-based Tablet such as the TabletKiosk Sahara tablet shown in Figure 5-1 [145]. This decision was based on its ability to be operated untethered (via an 802.11b network), with a finger rather than a stylus, its ability to run a full-featured commodity operating system (Windows XP), and it's generous display size. In all respects other than the size, the platform is meant to resemble a next-generation remote control that can be used comfortably on a couch or carried about the house from room to room. Even in terms of its size, it is not greatly more bulky than many high-end universal remote controls such as the Phillips Pronto [133] or the Sony Remote Commander [165], which both share with the Sahara the requirement that they be held with one hand or set on a stable surface while operated with the other hand. The larger screen size was selected in our case because it gave us the maximum flexibility to explore prototyping options with complete software reconfigurability (no hardware buttons were used as part of any of the prototype UIs).

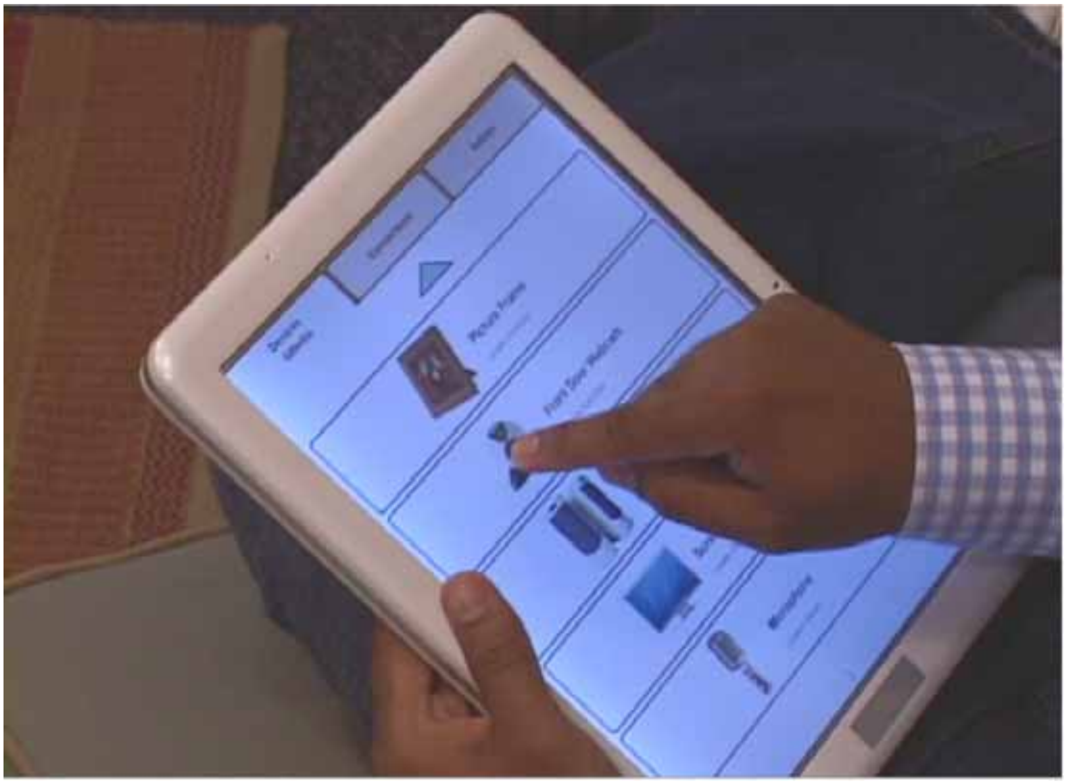


Figure 5-1: OSCAR is a handheld, touchscreen based application that allows users to discover, connect, and control multiple devices and services on a home network, as well as create re-usable compositions for quick access to commonly-used functions.

5.2 Designing OSCAR

OSCAR was designed using an iterative process consisting of four prototypes punctuated by evaluations. The first two iterations were followed by expert evaluations, and the second two were followed by user evaluations[8]. In the remainder of this chapter, I will describe the first three versions of the prototype—including the initial paper prototype, a medium-fidelity mock-up prototype, and the first interactive prototype. In addition, I will describe the results of the two expert evaluations. The results of the first user study, the design changes that were made for the second interactive prototype, and the results of the second user study are the subjects of Chapter 6.

Based on the overarching goals for the OSCAR project, including the desire to gain insight into the user experience goals of Obje, the desire to provide effective support for composition and control of home media networks, and the desire to study an embodiment of an end-user composition system, several of the basic design criteria for OSCAR were outlined before any of the design work began. In addition to the form factor decision described above, all versions also supported the same basic functions:

[8] While it would have been preferable to employ user evaluations during the early iterations as well, this turned out to be impossible due to issues with one of the funding agency's Human Subjects Review Board.

1) Browse, select, and control devices and services that have been automatically discovered on the network.
2) Connect compatible devices, services, and media sources, and then monitor, edit, and control the active connections.
3) Invoke, edit, and create reusable custom configurations of devices and media, which are labeled with user-provided names.

Given the particulars of the Obje service profiles (e.g., *DataSource*, *DataSink*, and *Aggregate*) and connection mechanisms described in Chapter 3, a "configuration" is intended to mean a description of a set of devices along with information about how particular *DataSources* should be connected to *DataSinks* via *TransferSessions*.

5.2.1 Issues with "Custom Configurations"

The fundamental challenge of the early phase of the design process was *how to best present the concept of custom configurations to users*. Two elements of the challenge, language choice and world view, were particularly difficult.

5.2.1.1 Language Choice

As I have already argued, custom configurations would be valuable to non-technical home users, but from the initiation of the design process, it was clear that the concept would be difficult to convey in understandable language. As we shall see, language to describe these configurations passed through several iterations, beginning with "Template," through "Recipe," and finally to "Setup."

5.2.1.2 World Views: Activities Versus Devices

Another question that became a focal point of the design process is how to organize the functions of the user interface to maximize ease of use as well as to communicate the range of capabilities of the application. The design focused on two approaches:

1) *Activities are primary*. The first screen that greets users upon encountering OSCAR is a list of pre-loaded compositions, representing different activities in which they can engage, along with a button allowing them to make a new composition. Users can by default interact with the list of compositions and if they discover that they needed functionality not currently represented in their list, they create a new composition to do what was needed. This approach is *activity-centered* in that users would see a list of compositions corresponding to activities that could be easily accessed and invoked. If the desired activity is not available, it would need to be configured before it could be invoked. However, it would remain available ever after. The advantage of the activity-oriented approach is that users are not forced to deal explicitly with devices in the normal case, rather they deal with a list of favorite compositions that are already pre-

defined to smoothly configure the network to carry out the activity described by their name.

2) *Devices and media sources are primary*. The first screen that greets users is the list of devices and media that are available. From this screen, the user can select a device or service and from there view and select from lists of compositions and active connections that included this device, and/or choose to use the device in a new composition. The advantage of the device/media-orientation is that it is easier to configure unscripted activities that are being carried out for the first time and gives the user a view of all of the devices currently available for use.

In both cases the primary goal of the interface is to allow users to make connections among devices and media. Also, in both cases, all three main functions (browsing devices, managing connections, and managing/invoking compositions) would be available. The difference between these two approaches would lie largely in the initial "home" screen that greets the user when first encountering or re-encountering OSCAR.

5.2.2 Personas and Scenarios

In order to guide the design process, I developed four Persona Households that captured a range of household characteristics. As with the traditional Persona-based design method [27, 57], the purpose of the Personas was to guide and provide coherence to the design process and give me a reference point outside of the designers' (i.e., my) own desires and experiences against which to gauge the likely effectiveness of design decisions. In terms of this particular process, the Personas also provided a richer picture for the expert review panel of OSCAR's intended users.

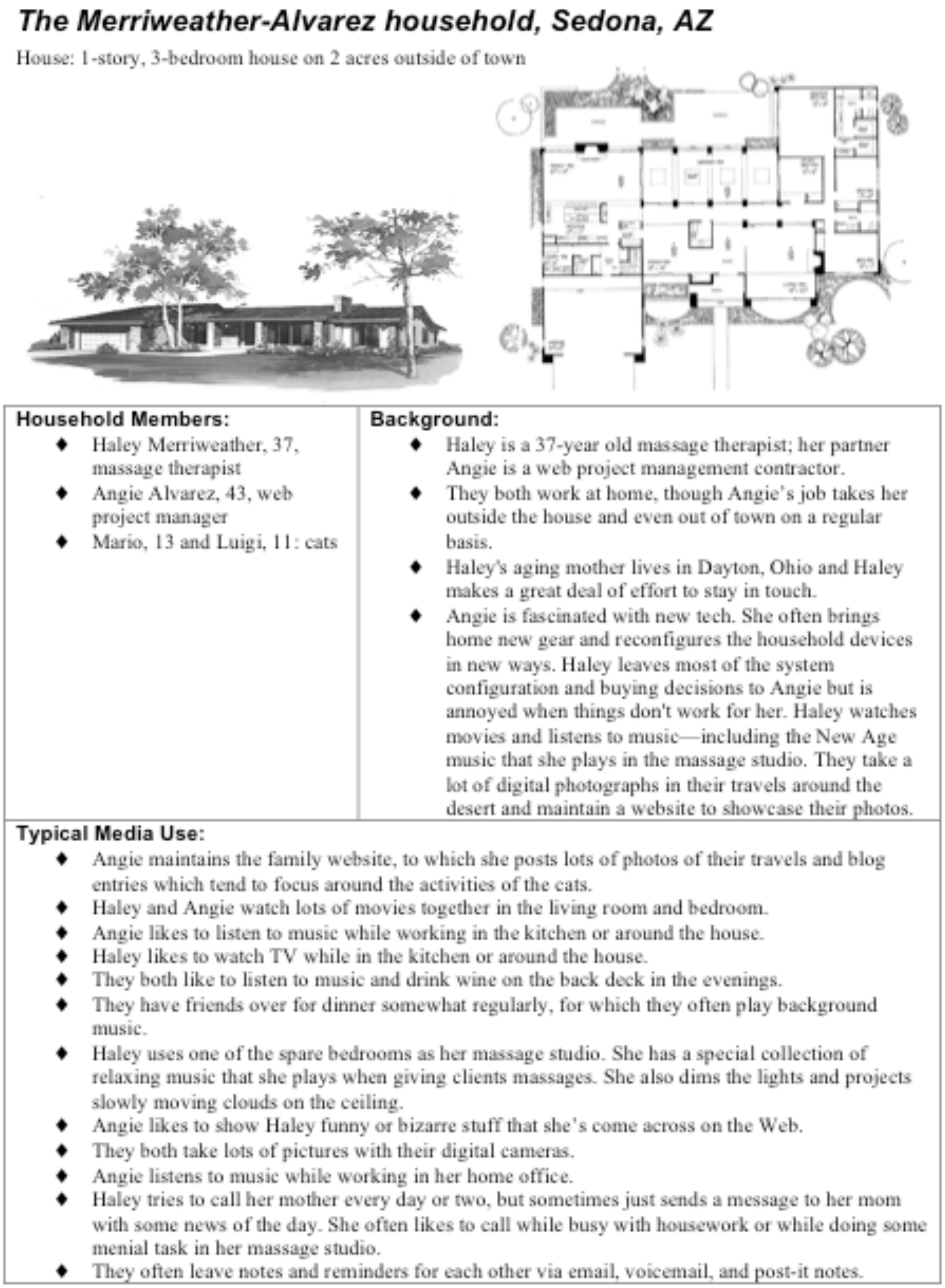
Based on the studies of home technology use discussed earlier in this chapter that suggest the household rather than the individual as the appropriate unit of analysis for domestic technology design, I modified the traditional notion of Personas in design [27] to include entire households. Since the home study in which I had participated represented only a narrow range of demographic types (relatively high-income, childless couples aged 30-45 in the Bay Area), I invented a set of households to guide OSCAR's design that was based more on personal experience and acquaintance. The four households were intended to span a range of ages, genders, living situations, house sizes, geographic regions, local environments, etc. One key finding of the home networking study [55] that was carefully reflected in the set of OSCAR personas was the uneven distribution of technical prowess and technology adoption preferences that was revealed to be the norm, even among the "early adopter" households that we studied. Therefore, I was sure to populate each household with different mixtures of inhabitants along each of these dimensions, including individuals fitting the different technology adoption profiles identified by Rogers [130].

Each Household Persona consists of a house photo or drawing, floor plan, house location, description of the occupants, and narrative about how the occupants typically interact with devices and media inside and outside the house.

Table 5-1 presents the basic information about each household, and Figure 5-2 gives an example of the type of information that was produced for one of the households. The full descriptions of the personas are included in Appendix B.

Name	Location	Occupants	Typical Media Use
The Engstroms,	Maple Grove, MN (newly built suburban development)	Dan Engstrom, 52, small business owner (auto parts distributor)	• Watch football games with friends • Show home movies & photos
		Michelle Engstrom, 50, schoolteacher	• Listen to radio • Watch cooking shows
		Audrey Engstrom, 17, high school junior	• SMS, email, phone with friends
		Bridgette Engstrom, 14, 8th grader	• Listen to music on headphones
		Duncan Engstrom, 11, 5th grader	• Play video games • Watch TV
		Butch, 3, golden lab	• None
The Merriweather-Alvarez household	Sedona, AZ (older house on 2 acres outside town)	Haley Merriweather, 37, massage therapist	• Play new age music in home massage studio
		Angie Alvarez, 43, web project manager	• Organize & publish digital photos
		Mario, 13 and Luigi, 11: cats	• None
Kari, Jesse, and Ruth's house	Durham, NC (3-bedroom rented house near Duke University)	Kari Weissbrun, 26, magazine editor	• Play World of Warcraft • SMS and email
		Jesse Amman, 26, waiter	• Download MP3s • Make mix CDs
		Ruth Sloan, 28, video artist/educator	• Video art projects • Maintain website
Eugene An's apartment	Seattle, WA (1-bedroom apartment in Capitol Hill)	Eugene An, 29, musician/administrative assistant	• Create music • Produce podcast
		(Frequent guest: Sabrina Lin, 27, social worker—his girlfriend)	• Watch DVDs • Listen to iPod while jogging

Table 5-1: An overview of the Household Personas used to help design OSCAR.



The Merriweather-Alvarez household, Sedona, AZ

House: 1-story, 3-bedroom house on 2 acres outside of town

Household Members:

- Haley Merriweather, 37, massage therapist
- Angie Alvarez, 43, web project manager
- Mario, 13 and Luigi, 11: cats

Background:

- Haley is a 37-year old massage therapist; her partner Angie is a web project management contractor.
- They both work at home, though Angie's job takes her outside the house and even out of town on a regular basis.
- Haley's aging mother lives in Dayton, Ohio and Haley makes a great deal of effort to stay in touch.
- Angie is fascinated with new tech. She often brings home new gear and reconfigures the household devices in new ways. Haley leaves most of the system configuration and buying decisions to Angie but is annoyed when things don't work for her. Haley watches movies and listens to music—including the New Age music that she plays in the massage studio. They take a lot of digital photographs in their travels around the desert and maintain a website to showcase their photos.

Typical Media Use:

- Angie maintains the family website, to which she posts lots of photos of their travels and blog entries which tend to focus around the activities of the cats.
- Haley and Angie watch lots of movies together in the living room and bedroom.
- Angie likes to listen to music while working in the kitchen or around the house.
- Haley likes to watch TV while in the kitchen or around the house.
- They both like to listen to music and drink wine on the back deck in the evenings.
- They have friends over for dinner somewhat regularly, for which they often play background music.
- Haley uses one of the spare bedrooms as her massage studio. She has a special collection of relaxing music that she plays when giving clients massages. She also dims the lights and projects slowly moving clouds on the ceiling.
- Angie likes to show Haley funny or bizarre stuff that she's come across on the Web.
- They both take lots of pictures with their digital cameras.
- Angie listens to music while working in her home office.
- Haley tries to call her mother every day or two, but sometimes just sends a message to her mom with some news of the day. She often likes to call while busy with housework or while doing some menial task in her massage studio.
- They often leave notes and reminders for each other via email, voicemail, and post-it notes.

Figure 5-2: An example of the information produced for one of the OSCAR Household Personas.

5.2.3 First OSCAR Prototype: Paper Prototype

The first OSCAR prototype was the paper prototype shown in Figure 5-3. The initial goal of the paper prototype was to quickly design a user interface and test it with users. The idea behind a paper prototype is to prototype, as completely as possible, the entire behavior of an application before writing a single line of code [124, 137]. The advantages of using a paper prototype over code are numerous: the investment in any particular design decision is much less, so the costs of abandoning a design are very low; users and reviewers feel more comfortable expressing negative opinions about the prototype because they perceive it as being less finished and therefore more amenable to modification; exploration of new ideas and functionality can be carried out instantaneously, even during the course of a usability test or expert review—users and reviewer can see examples of their suggestions within seconds in order to see if their feedback has been interpreted correctly; and the list goes on.

A paper prototype can be used for usability tests and/or expert walkthroughs by having one of the test administrators act as the “computer.” Whenever the user/expert carries out some action the “computer” reconfigures the paper prototype to show the results of the user’s action. Such a prototype is effective at catching coarse usability problems, such as architecture and navigation errors, that are important to capture early in the design process. Clearly there are other classes of error, such as software bugs, screen layout problems, and responsiveness issues that are not turned up by such early prototypes. Paper prototypes are one tool in a suite of tools available to UI designers and the experience of many usability professionals has shown that they are an extremely effective tool in the early design phases.

The OSCAR Paper Prototype consists of two main UI screens whose contents adapt to allow users to accomplish various tasks related to service composition and execution. As can be seen in Figure 5-4 through Figure 5-8, the prototype was designed as a pair of background screens and a set of overlaid content components that could be added and removed to show different states of the UI.

The first screen a user sees is the “Template Selection” screen (shown in Figure 5-4)—which allows the user to browse and select a template (i.e., a configuration description) that she can then activate or edit. The second screen is the “Template Interaction” screen, which allows a user to actuate, modify, or control a single template. Figure 5-5 shows a simple template that is currently inactive. If the user wishes to interact further with this template, there are a number of options. As shown in Figure 5-6, the user can replace the source or destination component by browsing for a replacement. For the source in particular, the user can choose either a particular source component or a collection of them. When satisfied with both endpoints of the connection, the user can tap the center arrow to change its state. If it is currently idle, as indicated by the dotted arrow, the connection will be started and OSCAR will provide control options relevant to the current connection, as shown in Figure 5-7. To further configure the template's behavior when it is activated in the future, the user can tap the gear icon underneath either the source or destination slot, or underneath the arrow. Figure 5-8 shows an example of what happens when the user taps the gear underneath the destination slot. There are other options available, as well, and some will become clear as I describe the walkthrough version of the paper prototype that was produced in preparation for the first expert evaluation.

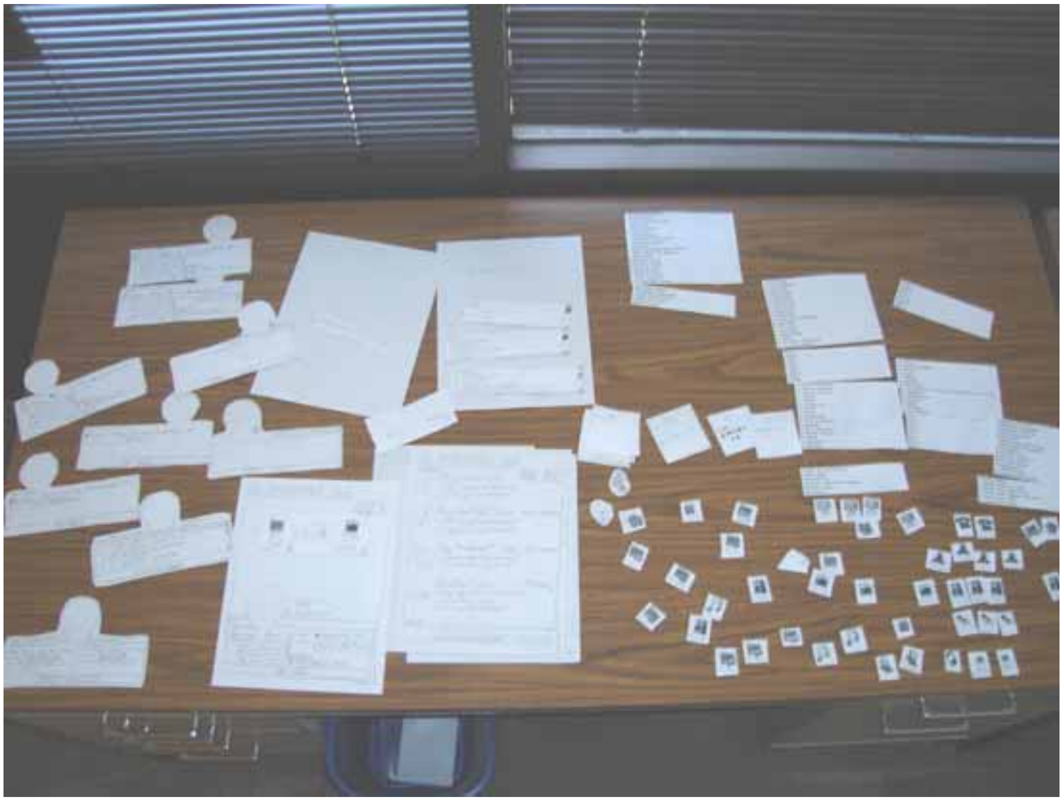


Figure 5-3: The entire Paper Prototype, including the major screens and many "widgets" representing contents of the screens that a user is expected to encounter during the course of carrying out the usability test tasks.

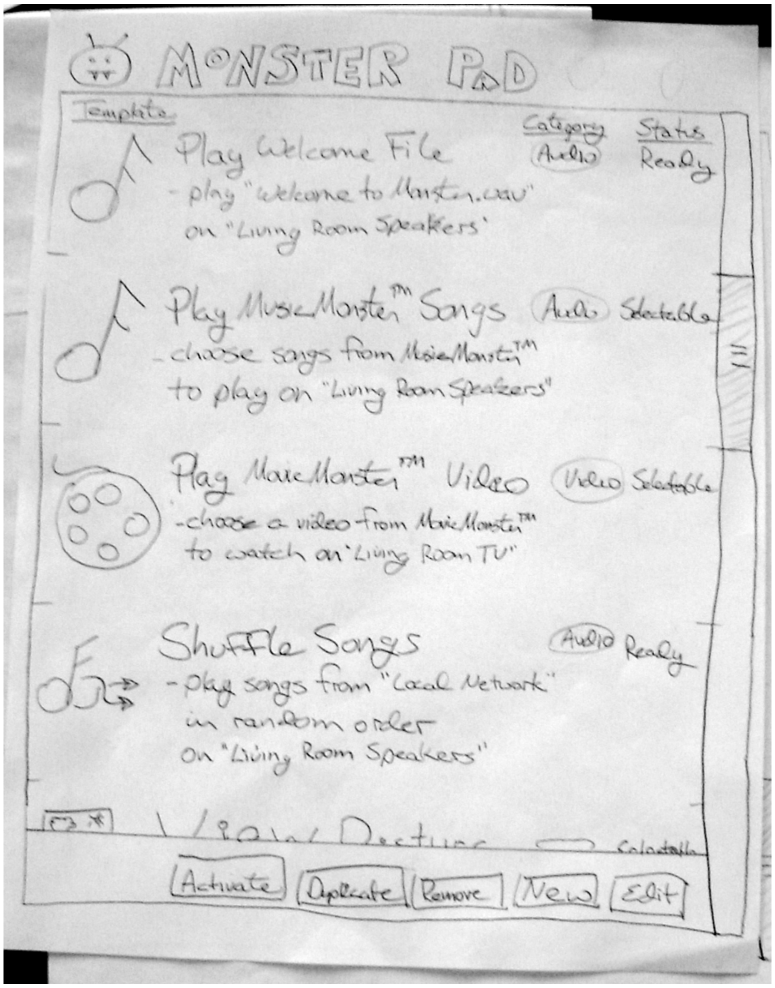

MONSTER PAD
Template
Category
Status
Play Welcome File
Audio
Ready
- play "Welcome to Monster.wav"
on "Living Room Speakers"
Play MusicMonster™ Songs
Audio
Selectable
- choose songs from MusicMonster™
to play on "Living Room Speakers"
Play MovieMonster™ Video
Video
Selectable
- choose a video from MovieMonster™
to watch on "Living Room TV"
Shuffle Songs
Audio
Ready
- play songs from "Local Network"
in random order
on "Living Room Speakers"
Selectable
Activate
Duplicate
Remove
New
Edit

Figure 5-4: The main screen that greets a user of the initial OSCAR paper prototype. A list of templates is presented, from which the user can select one to "activate," "duplicate," "remove," or "edit." Also the user can choose to create a new Template from scratch.

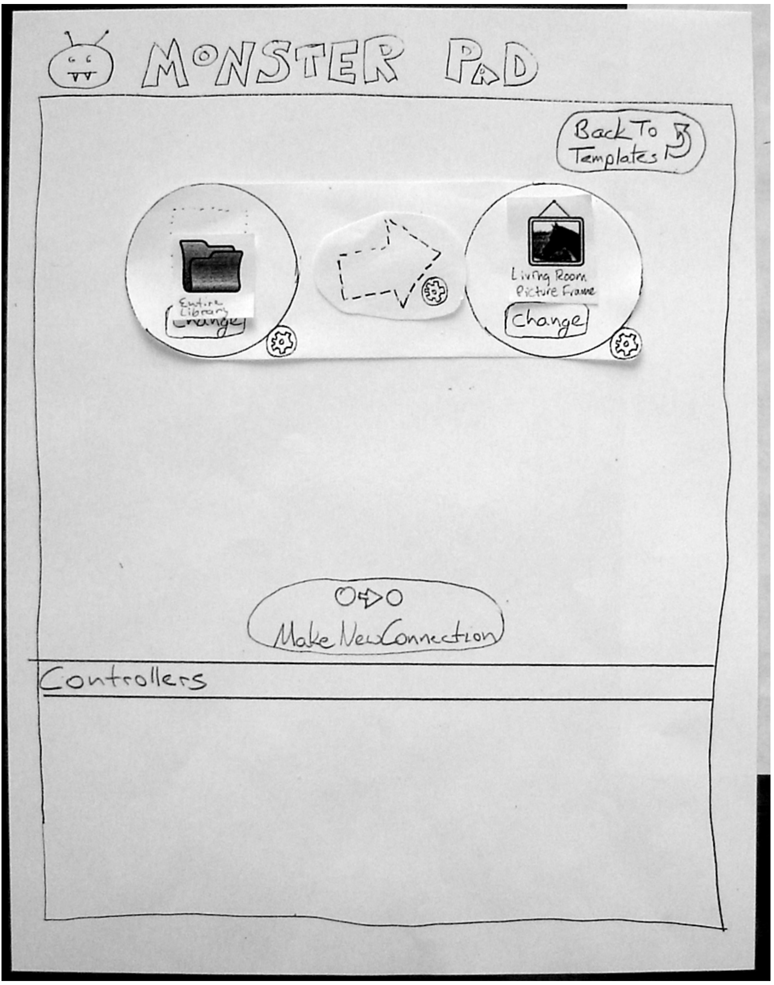


Figure 5-5: The "Template Interaction" screen showing a currently inactive Template (as evidenced by the dotted arrow). The template depicted consists of one connection, whose source is the "Entire Library" of photos and whose destination is the "Living Room Picture Frame." If the user tapped the arrow, the connection between the Photo Library and the Picture Frame would be established, and specific photos would be chosen for display based on the advanced settings that could be accessed via the gear icon to the lower right of the arrow.

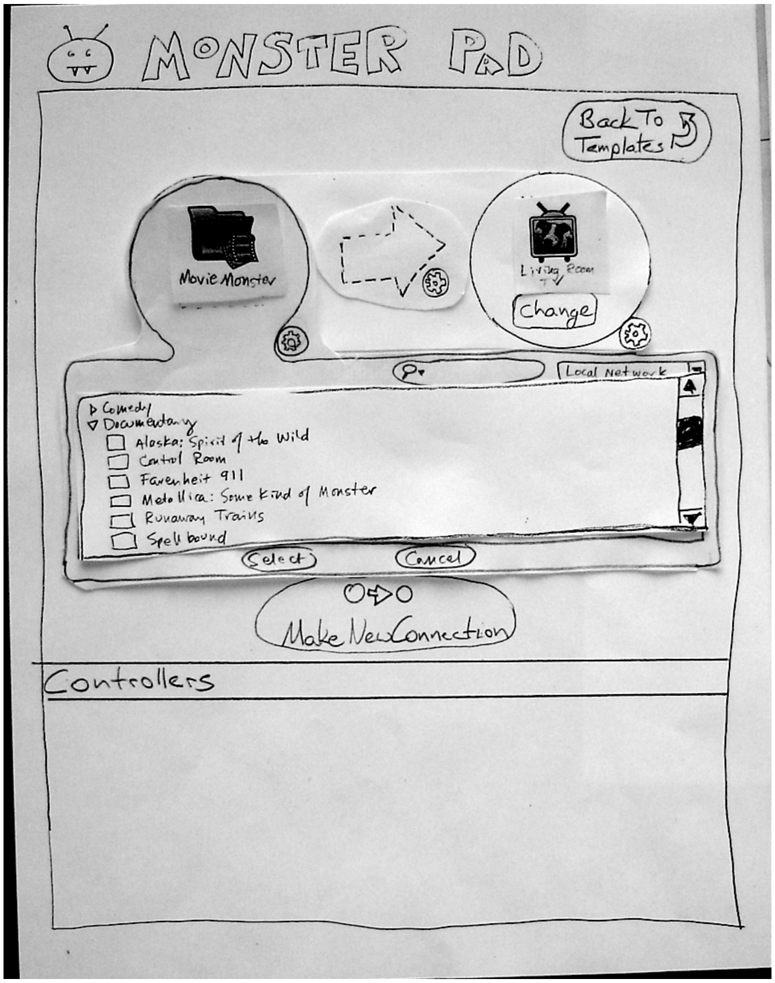


Figure 5-6: The "Template Interaction" screen while the user is selecting a particular source for a connection from an Aggregate containing multiple possible sources.

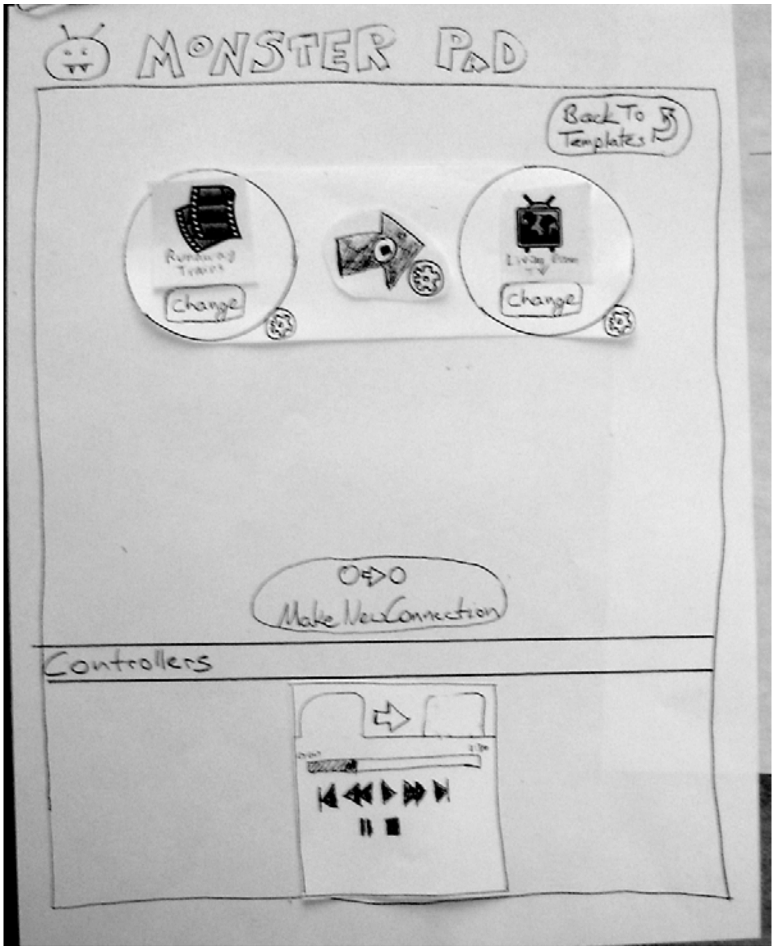
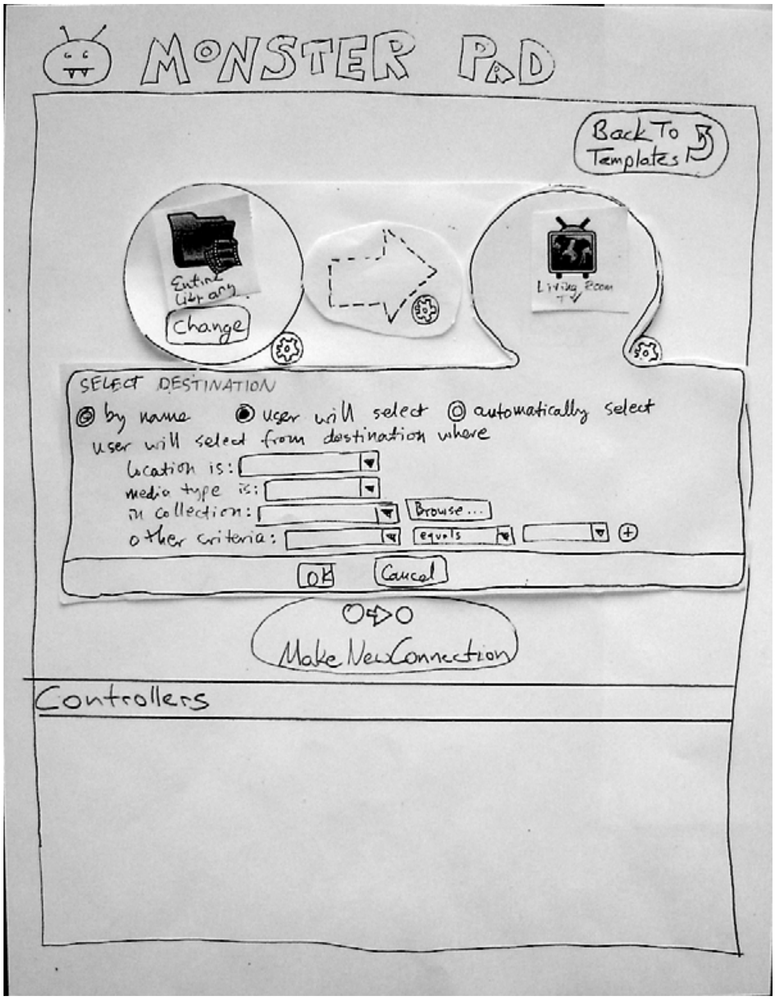


Figure 5-7: The connection is active and the user has access to controls specific to controlling this connection—in this case the controls allow a user to play, pause, rewind, etc. a video stream that is playing on a TV.



SELECT DESTINATION

○ by name ● user will select ○ automatically select

user will select from destination where

location is: [▾]

media type is: [▾]

in collection: [▾] [Browse...]

other criteria: [▾] [equals ▾] [▾] ⊕

[OK] [Cancel]

Figure 5-8: The "Template Interaction" screen after the user has elected to configure the rules for selecting a destination.

5.2.3.1 Preparing for the Expert Evaluation: Walkthroughs and "Screenshots"

As mentioned previously, although the OSCAR paper prototype was initially designed to be used for early user studies, circumstances were such that I was unable to conduct those studies and instead adapted the design plan to include an early expert evaluation.

In preparation for the first expert evaluation, the elements of the paper prototype were all scanned in and re-assembled digitally to create a set of walkthrough screenshots. Fifty-two screenshots were created, corresponding to the steps a user would take in order to accomplish four tasks:

Task 1) Play the "Welcome" file on the living room speakers (4 steps).

Task 2) Locate the documentary "Runaway Trains" on the MovieMonster service and begin watching it on the living room TV. (8 steps)

Task 3a) Create a template that causes the picture frame by the front door to display a new image from your photo library every time you activate the template. (12 steps)

Task 3b) Now modify the template so that a new photo is displayed every 10 minutes (7 steps).

Task 4a) You read in a review of the MediaMonster that it was possible to create a template that causes your music to follow you around the house. Create that template. (12 steps)

Task 4b) Now modify the template so that your pictures *also* follow you around the house. (9 steps)

In addition to the walkthrough screenshots, some additional information about the design, its intended audience, and its envisioned use were included in a set of instructions provided to the reviewers. The (digitized) paper prototype walkthrough created for the first expert evaluation was derived directly from the paper prototype itself, and many of the screens were created by hand from the components of the paper prototype and scanned for sharing digitally. In addition, a number of the screenshots were manipulated after scanning, or contained elements that had been copied and pasted from other screens that had been previously scanned. The resulting walkthrough did not differ in significant ways from the initial paper prototype, but some details were fleshed out in the walkthrough that had been left unaddressed in the paper prototype. The following paragraph describes part of the interaction sequence for Task 3 as it was represented in the paper prototype walkthrough.

In Figure 5-9, the user has selected the template "View Picture on Frame" from the list of pre-installed Templates because it appears to be the closest in functionality to what they ultimately want to achieve. They then press "Activate" to view the template details (in this case, "Edit" would have been the more appropriate choice, but this walkthrough demonstrates that the design is somewhat robust to minor errors). After pressing "Activate," the user is presented with Figure 5-10, which is prompting the user to select a photo for display. Since the goal is not, in fact, to display a photo right this instant, but to create a template that automatically selects a photo when the template is activated, the user ignores the browse dialog and elects to configure the template by tapping the gear icon underneath the source slot. Next, the screen shown in Figure 5-11 is displayed, which presents a set of configuration options for the source slot. The user chooses the "automatically select" radio button and chooses "image" from the list box next to the label "media type is," and finally taps "Browse…" to the right of the label "in Collection." This results in the pair of dialogs shown in Figure 5-12. After making further choices in this dialog and saving them by tapping "Select," the user is presented with the screen shown in Figure 5-13, on which (s)he taps "Choose" under the "Picture Frame" icon. This results in a second browse dialog that shows components that are compatible with the "media type is: image" selection made when configuring the source slot. After choosing "Front Door Picture Frame" on the screen shown in Figure 5-14 and tapping the "Select" button, the user then taps the center arrow to activate the template. The resulting screen, shown in Figure 5-15, provides the user with controllers that are specific to the picture frame and the image renderer, along with status information and stop/start/change controls related to the connection as a whole.

The walkthrough just presented covers most of the functionality available in the OSCAR paper prototype. We have seen how a user can

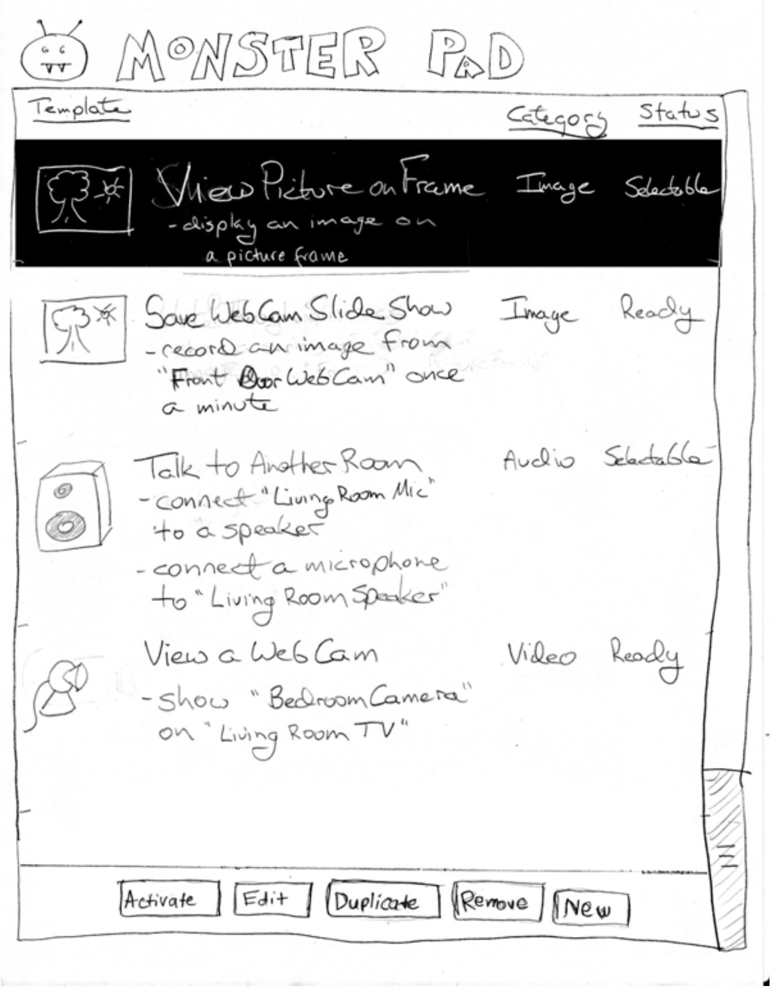
- View and select from the list of templates
- Browse the list of devices and media that are available on the network
- Incorporate those devices and media into templates and connections
- Edit a template to define rules regarding how selected devices and media are connected at template activation time
- Activate a template and view the resulting connection(s)
- Gain access to controls for stopping and starting connections
- Gain access to device- and media-specific controls for an active connection

The OSCAR paper prototype was designed to afford a handful of additional capabilities that were not highlighted by the foregoing walkthrough. Additional options include:

- Configuring a template to prompt the user to select from a set of options when the template is activated
- Configuring a template to automatically select a new item from the list of sources at a particular time interval (slideshow mode)
- Configuring a template to automatically select a new item from the list of sources upon completion of the current source's connection (playlist mode)

- Defining custom criteria to generate the list of source or destination components
- Creating a template that includes multiple connections

All of these additional functions were included in the full set of walkthroughs that were delivered to the expert review panel.



Template	Category	Status
View Picture on Frame - display an image on a picture frame	Image	Selectable
Save WebCam Slide Show - record an image from "Front Door WebCam" once a minute	Image	Ready
Talk to Another Room - connect "Living Room Mic" to a speaker - connect a microphone to "Living Room Speaker"	Audio	Selectable
View a WebCam - show "Bedroom Camera" on "Living Room TV"	Video	Ready

Figure 5-9: The screen from the initial OSCAR paper prototype viewed by a user as they attempt to carry out Step 3 of Task 3 ("Create a template that causes the picture frame by the front door to display a new image from your photo library every time you activate the template"). In the previous step, the user selected the template "View Picture on Frame." In Step 3, the user will tap the "Activate" button at the bottom of the screen.

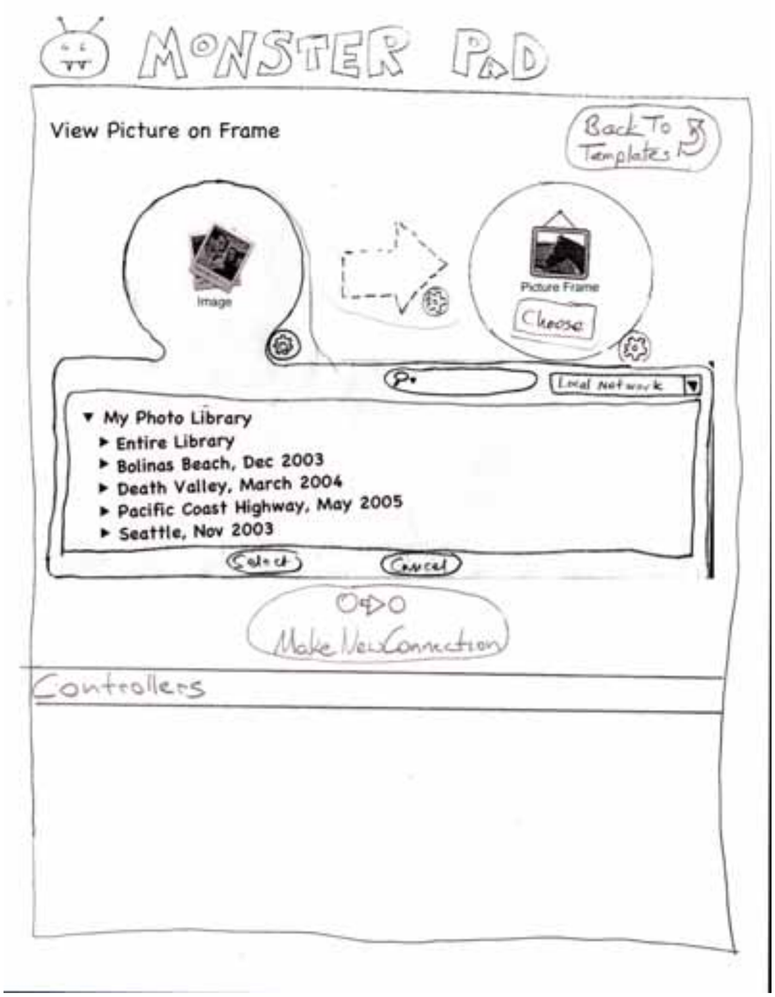
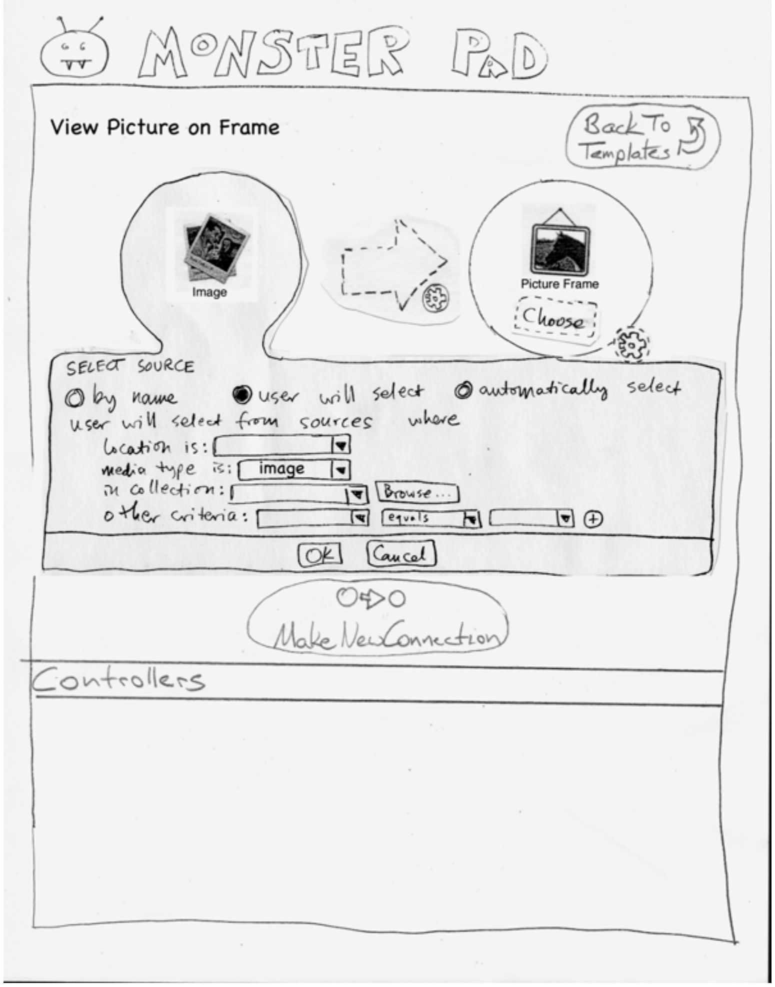


Figure 5-10: After pressing "Activate" on the screen shown in Figure 5-9, the user is prompted to select a photo.



○ by name ● user will select ○ automatically select

user will select from sources where

location is: [▾]

media type is: [image ▾]

in collection: [▾] [Browse...]

other criteria: [▾] [equals ▾] [▾] ⊕

Figure 5-11: After selecting a photo, the user taps the "gear" icon below the source slot, which results in this screen being shown.

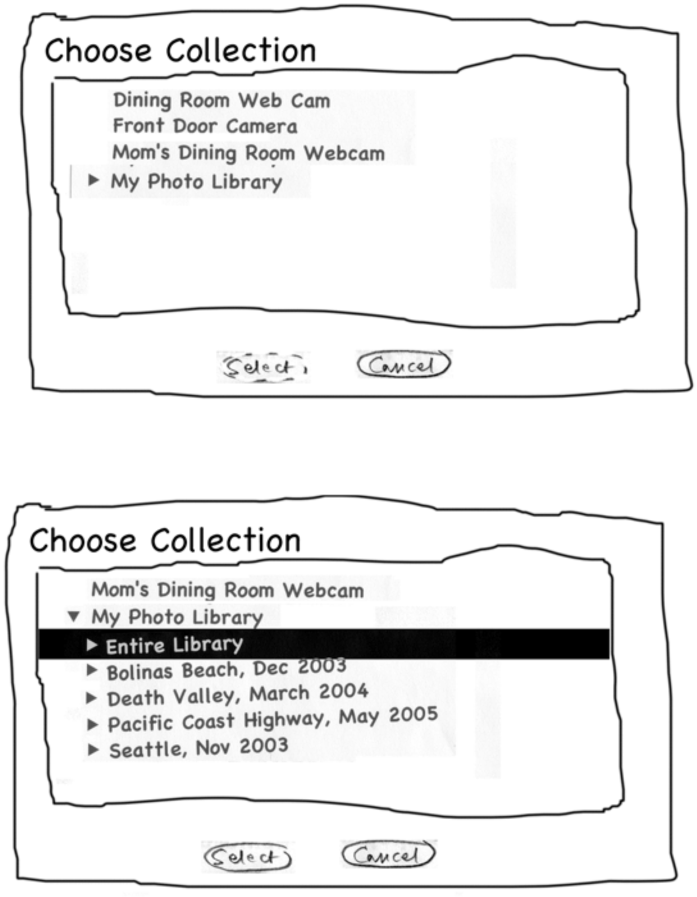


Figure 5-12: The user wishes to choose a collection from which photos will be selected for display when this template is activated. After pressing "Browse…" to the right of the label "In collection" on the screen shown in Figure 5-11, this screen is shown.

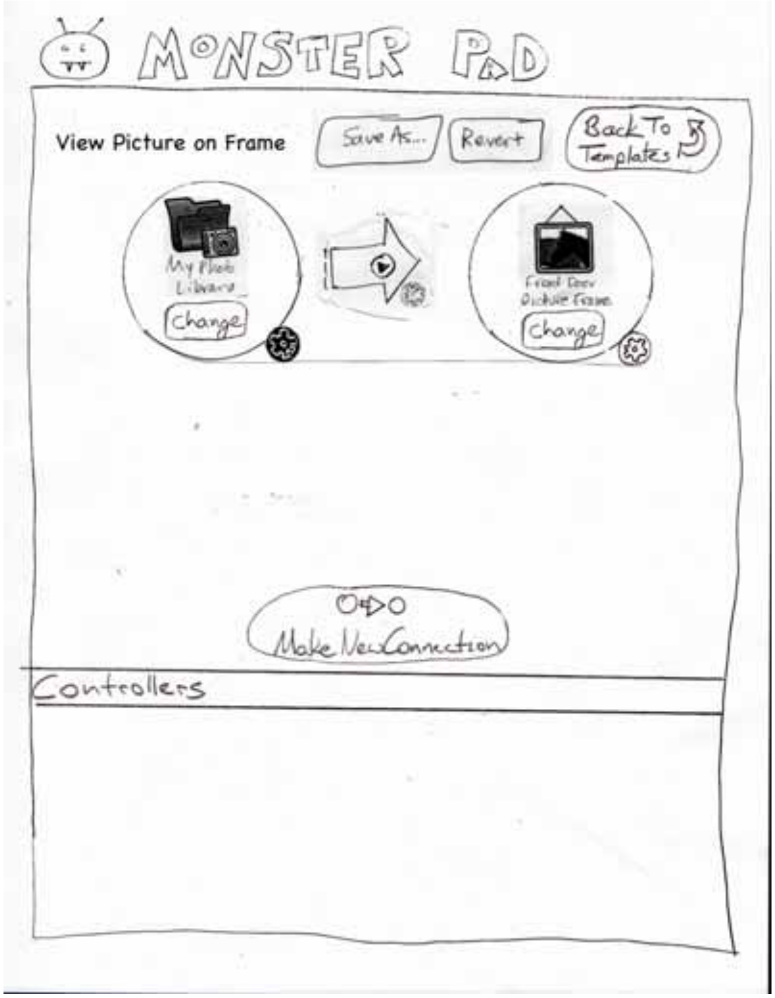


Figure 5-13: Once configuration of the source slot is complete, the current state of the connection is shown.

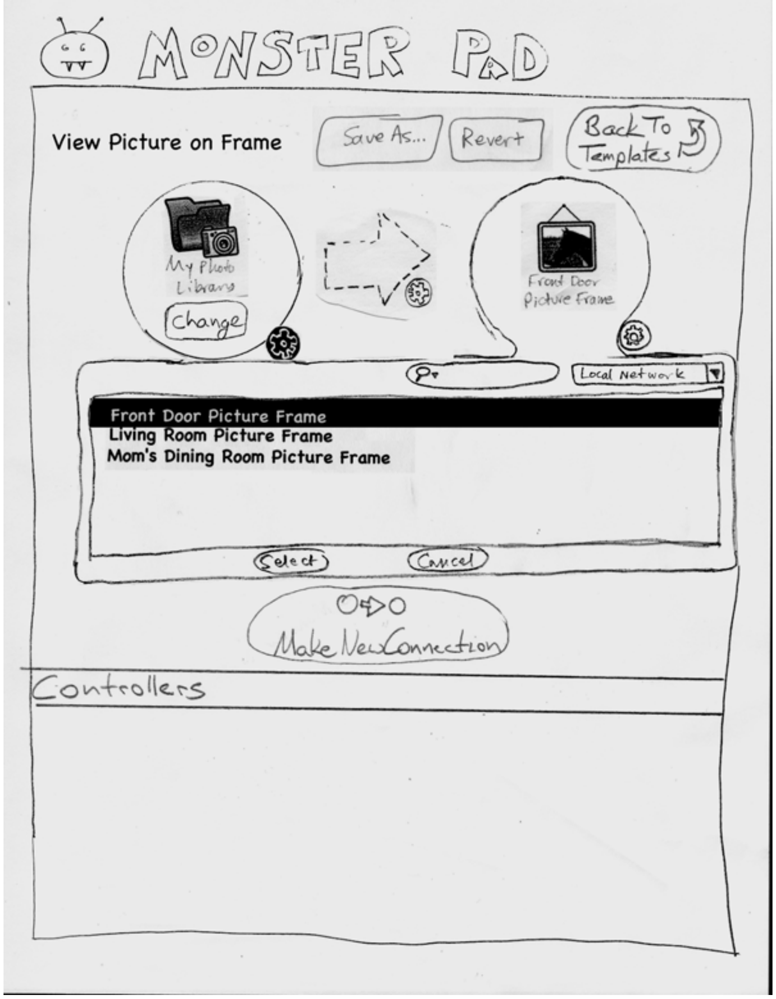


Figure 5-14: In order to choose a destination for this template's connection, the user taps the button "Choose" under the destination icon in Figure 5-13. This screen is the result.

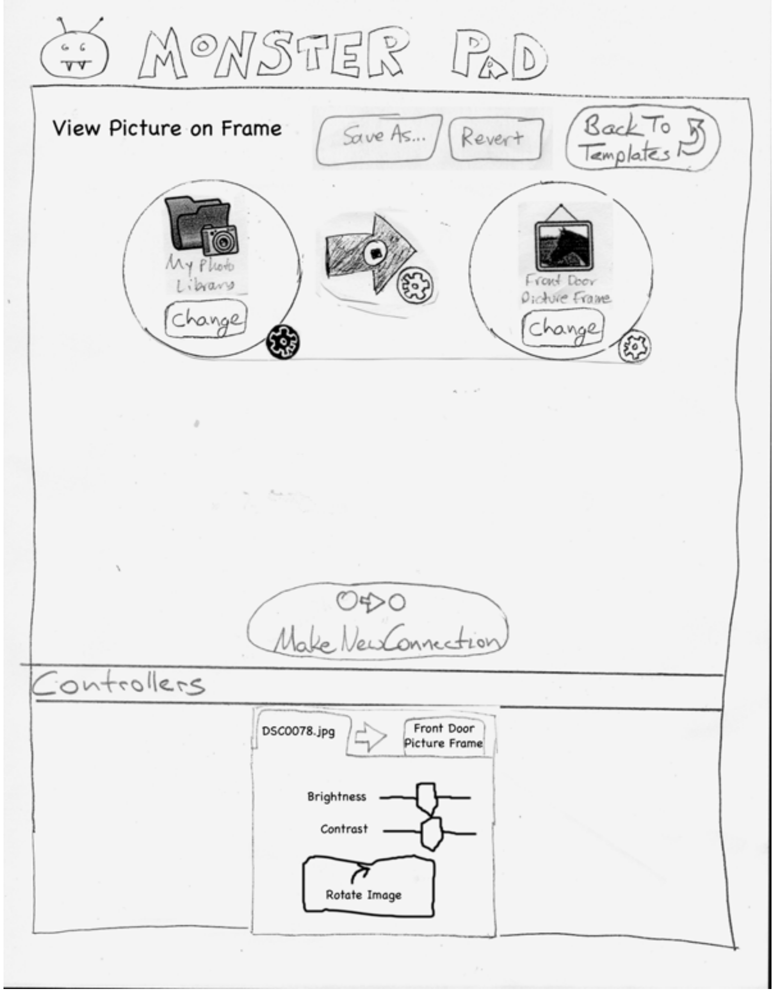


Figure 5-15: Finally, after completing the configuration, the user activates the template. This screen shows the active template with one connection that can be stopped and started using the arrow in the center. Also device-specific controls are shown along the bottom.

5.2.4 First Expert Evaluation

For the two expert reviews of OSCAR prototypes, I recruited a panel of four usability experts from within PARC who were not affiliated with the OSCAR project. In addition to being well versed in usability and user interface design, they were all reasonably avid consumers of both analog and digital media, which qualifies them as domain experts as well for the purposes of this application.

In preparation for each expert evaluation, I prepared a detailed walkthrough of the OSCAR design and delivered the walkthrough along with instructions and other supporting materials (e.g., the Persona descriptions) to the panel of experts. Each expert spent a number of hours over several days reviewing the prototype and applying their expertise to assess the usability of the application being designed. Each reviewer produced a report of the usability issues identified during their evaluation, and each usability issue was given a severity rating of high, medium, or low. In addition, they were asked to comment on positive aspects of the design that they felt should not be changed, and they were also asked to provide high-level comments about the proposed tasks, personas, usage scenarios, form factor, etc. After receiving all four reports, I combined the feedback into a single, non-redundant, prioritized list of issues and suggested improvements.

The feedback from the expert evaluations was overall positive in terms of the utility, general direction, envisioned tasks, and intended users of OSCAR, but there were numerous issues both large and small that were identified by the experts.

First there were a handful of areas where the evaluators felt the UI was effective. Three were pointed out by multiple evaluators: the visual representation of connections between devices, the linking of connection/template slots to the dialogs that allow users to browse for the slots' contents, and the visibility of the range of configuration options when editing or creating a template. These elements are highlighted in Figure 5-16.

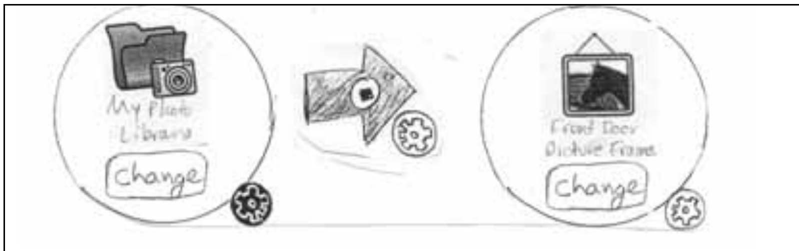
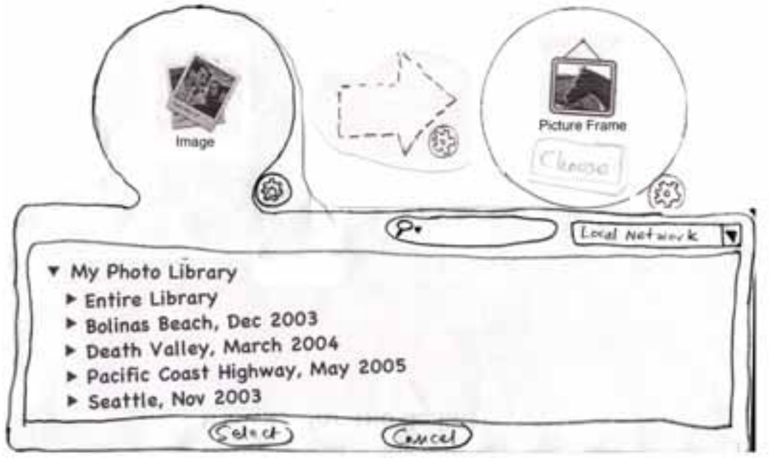
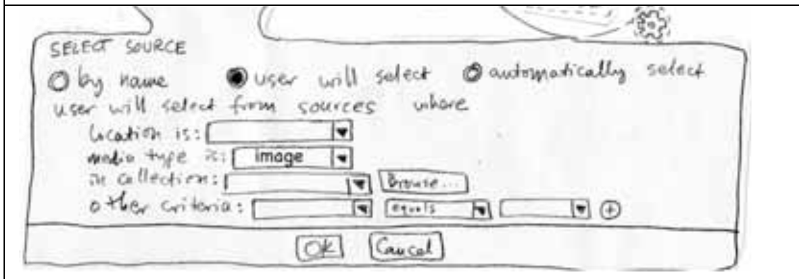
 My Photo Library Change Front Door Picture Frame Change	Visual representation of connections
 Image Picture Frame Choose Local Network Select Cancel ▾ My Photo Library ▸ Entire Library ▸ Bolinas Beach, Dec 2003 ▸ Death Valley, March 2004 ▸ Pacific Coast Highway, May 2005 ▸ Seattle, Nov 2003	Linking of browse dialog to “slots”
 SELECT SOURCE ○ by name ● user will select ○ automatically select user will select from sources where location is: media type is: image in collection: Browse... other criteria: equals 	Communication of range of configuration options

Figure 5-16: A few elements of the initial OSCAR paper/walkthrough prototype were called out by the panel of experts as being valuable design elements. Three of those design are represented here in the left column, with the aspect of the depicted design that was called out by the panel listed in the corresponding right column.

In all, the evaluators identified 151 discrete usability issues, 31 identified as "highly severe." Several of the severe issues addressed aspects of the system as a whole rather than the OSCAR UI in particular, while others were fairly specific about elements of the UI.

Some of the issues that were identified were determined to be beyond the scope of the OSCAR project. For example, some of the panelists wondered how OSCAR would deal with resource contention, i.e., two members of the household wishing to use a particular device (say, the living room speakers) at once. Others were concerned about the boundaries between public and private devices in multi-person homes, and worried about how users would manage their sharing policies. One panelist in particular was both concerned and excited by the possibility that a system like OSCAR could allow one user to gain awareness of the presence and activities of other users, whether in the same or in different households. While each of these concerns and/or opportunities are interesting and present opportunities for future research, it did not seem wise to expand the scope of OSCAR to address these issues directly, but rather to keep the focus on how and end-user will configure and control sets of devices for themselves and for their housemates. Other issues deemed out-of-scope included support for controlling/monitoring devices currently not visible (e.g., in another room or another household), and the scalability of the template and device lists to handle large numbers (100s or more) of items.

In terms of usability issues that directly affected the established goals for OSCAR, there were two areas around which the panelists identified multiple high-priority issues. These were clearly the areas most in need of improvement: the terminology and models surrounding Templates, Connections, and their relationship to each other; and the amount of information and context provided while browsing for components (especially when trying to fill a source or destination slot for a template/connection).

The second prototype was designed to address the critical issues as well as many smaller issues. In particular, it was clear from the expert evaluation that the conceptual model of the interface and the terminology that is used to reflect that conceptual model needed a significant overhaul. The second prototype focused on improving these issues, and in so doing also addressed many of the smaller issues—for example several troublesome elements of the UI were removed and replaced with new options that better reflected the revised conceptual model.

5.2.5 Second OSCAR Prototype: Medium Fidelity Mockups

I created a second prototype based on the feedback from the expert panel. This prototype was rendered in a "medium-fidelity" fashion, i.e. it is a set of computer-drawn screen shots that are not interactive. The second prototype was dramatically redesigned to address the primary concerns of the reviewers. Five major modifications were made:

1) Tab-based global navigation was introduced, with each major function of the UI (“Connections,” “Recipes,” “Ingredients,” and “Events & Schedules”) separated into its own tab. In addition a customizable “Home” tab was provided[9].
2) “Templates” were renamed to “recipes,” and accompanying language was used as well (e.g., devices and media were dubbed “ingredients” in order to reinforce their relationship and use as components of recipes).
3) Templates/recipes were separated from connections, and each placed in a separate section of the UI, delineated by separate top-level tabs.
4) Devices and media (a.k.a. “Ingredients”) were now also given their own top-level tab, so that users could view and interact directly with all of the resources available on their networks.
5) The UI for browsing and selecting devices from within the template/recipe and connection editing pages was improved to provide greater detail about each device and component, such as the component’s icon and location.

The re-casting of templates as recipes represented a particularly important shift in the design, and so I will briefly describe the thinking behind the new language choice.

[9] The functionality of the “Home” and “Events & Schedules” tabs were never fully resolved, and they were eventually dropped from OSCAR.

5.2.5.1 The Recipe Metaphor

In the first low-fidelity prototype we used the term “template” for custom configurations, and presented them as shown in Figure 5-4 through Figure 5-8. Our expert review panel gave us feedback that the term “template” was intimidating, confusing, and ambiguous. They especially thought that users of Microsoft Office would have prior, perhaps negative, associations about the meaning, capabilities, and usability of “templates.” I agreed with their assessment and sought a new term.

After considering many alternatives, I settled on the term “recipe” to express the notion of custom configurations. The recipe concept appears to fit quite nicely with the concept of custom configurations. A recipe is generally understood to be a kind of template: it can be doubled or halved, and substitutions are allowed. Like OSCAR’s configurations, recipes consist of ingredients that are combined at some point in time to create something that is qualitatively different than the sum of the parts. There may be many different recipes that produce the same dish, just as there may be many different configurations that support the same basic activity, such as “listen to music.” In addition, the term “recipe” evokes images of domesticity that hint at the historical dichotomy of the home as a place of both leisure and work. Recipes can be shared among friends and family members, and can be created, modified, and prepared by experts and novices alike.

OSCAR's customizable, re-usable compositions were therefore renamed "recipes."

5.2.5.2 Preparing for the second evaluation: Walkthroughs

The format of the walkthroughs for the second evaluation was the same as that used for the first evaluation: a set of screenshots corresponding to the steps that a user would take to carry out a set of tasks. The tasks were modified somewhat from the tasks used in the first iteration to reflect a clearer understanding of the emerging functionality of OSCAR and a greater emphasis on the Personas.

Task 1) Play Welcome to MonsterPad (8 steps).

- Dan Engstrom has just purchased the MonsterPad.
- He plays the "Welcome to MonsterPad!" recipe to test his setup.
- When he's done he deletes the recipe from his list of recipes.

Task 2) Watch a Netflix movie (10 steps).

- Haley uses the "Watch a Netflix Movie" recipe to start watching the documentary Runaway Trains.

Task 3) Show off Photos (15 steps).

- Dan creates a new recipe to help him show off photos (on the living room TV) when guests come over.

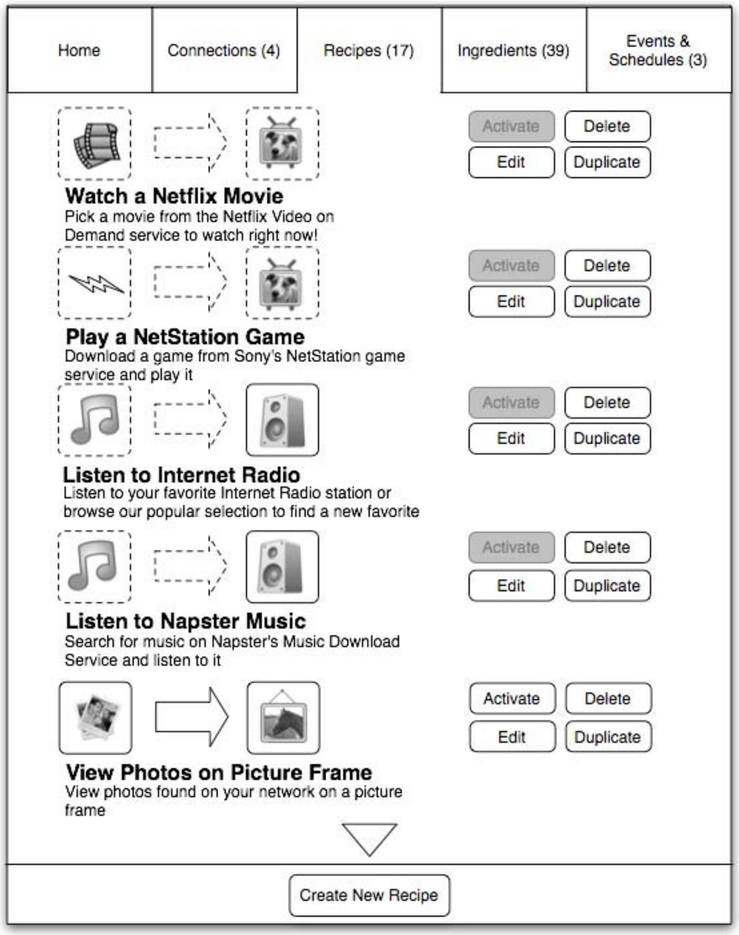
Task 4) View recent photos on desk frame. (17 steps).

- Angie creates a recipe that shows photos (from anywhere on the network) that were added in the past two days on her desktop photo frame.
- When she's done, she activates the recipe.
- Then she stops Runaway Trains, which is playing on the Living Room TV.

Figure 5-17 through Figure 5-23 shows a subset of the screens for the Task 3 walkthrough.

In Figure 5-17, Dan sees the list of current recipes, decides to create a new one by tapping "Create New Recipe." He sees a screen shown in Figure 5-18, inviting him to configure the new recipe. He first configures the source slot by choosing "Search for ingredients when this recipe is activated," and then proceeding to make selections in each of the sub-dialogs for "scope," "filter," "sort," and "select." Four steps later, the screen in Figure 5-19 is presented. He chooses to select a specific component, the Living Room TV, for this recipe's destination, so he taps "Choose" and is presented with the browse dialog in Figure 5-20. After choosing the Living Room TV and viewing the results of his actions in Figure 5-21, he taps in the "Recipe Name" text box near the top of the screen and is presented with the soft keyboard shown in Figure 5-22. Finally, after giving the recipe a name and saving his work, he returns to the list of recipes in Figure 5-23 and sees his new recipe in the list. At this point he can activate, edit, delete, or duplicate his new recipe.

Figure 5-24 through Figure 5-27 show additional functionality not demonstrated in the foregoing walkthrough. The OSCAR medium fidelity prototype also provides users with the ability to monitor and control multiple connections (Figure 5-24), view details and additional controls on a specific connection (Figure 5-25), browse and select from the list of all available devices and media (Figure 5-26), and view details and controls for a specific device or media source (Figure 5-27).



Watch a Netflix Movie
Pick a movie from the Netflix Video on Demand service to watch right now!

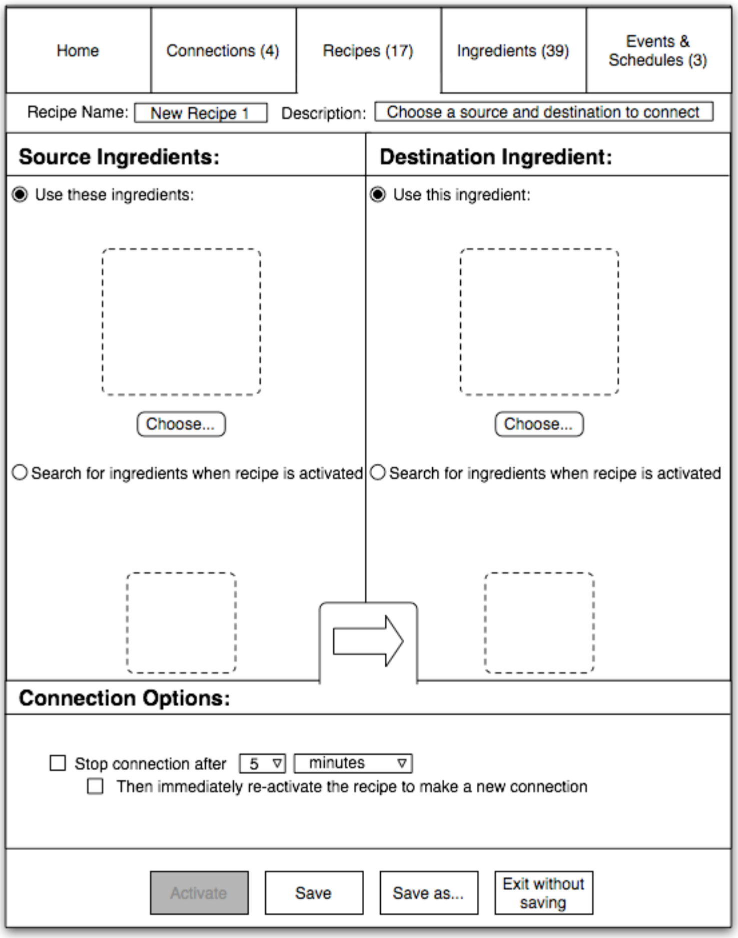
Play a NetStation Game
Download a game from Sony's NetStation game service and play it

Listen to Internet Radio
Listen to your favorite Internet Radio station or browse our popular selection to find a new favorite

Listen to Napster Music
Search for music on Napster's Music Download Service and listen to it

View Photos on Picture Frame
View photos found on your network on a picture frame

Figure 5-17: The list of recipes as they appeared in the OSCAR medium-fidelity prototype. From this screen the user can activate, edit, delete, or duplicate any existing recipes, or can create a new one.



Recipe Name: New Recipe 1 Description: Choose a source and destination to connect

Source Ingredients:

● Use these ingredients:

○ Search for ingredients when recipe is activated

Destination Ingredient:

● Use this ingredient:

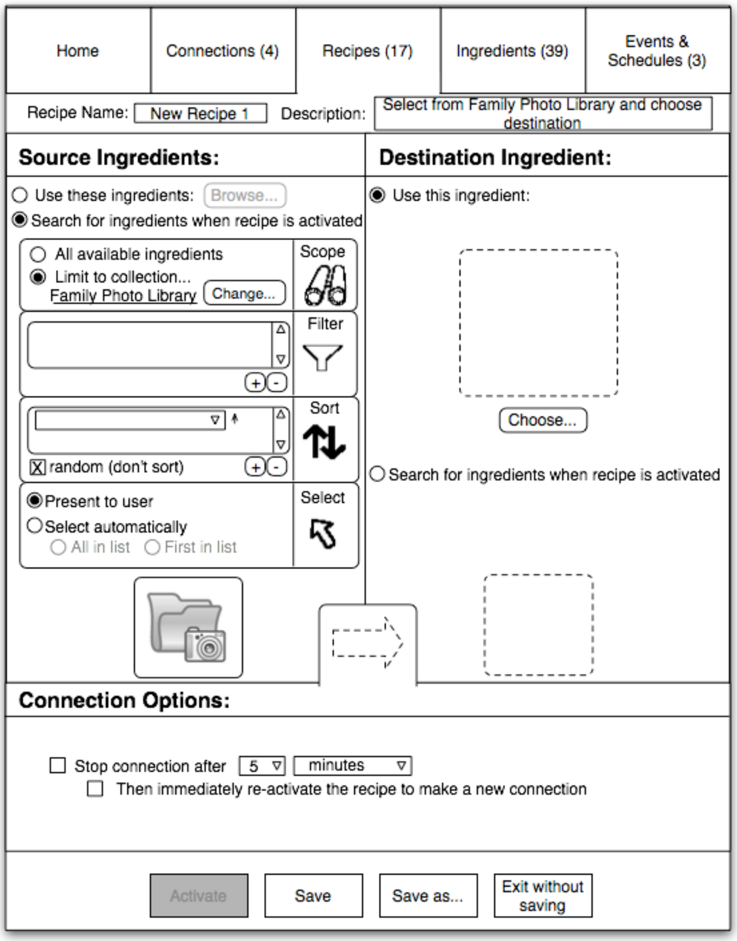
○ Search for ingredients when recipe is activated

Connection Options:

□ Stop connection after 5 ▿ minutes ▿

□ Then immediately re-activate the recipe to make a new connection

Figure 5-18: The screen shown after a user has elected to "create new recipe." Both source and destination slots are undefined as indicated by the dotted outlines and the fact that the "Activate" button is disabled.



Recipe Name: New Recipe 1 Description: Select from Family Photo Library and choose destination

Source Ingredients:

○ Use these ingredients:

● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection...

Family Photo Library

☒ random (don't sort)

● Present to user

○ Select automatically

○ All in list ○ First in list

Destination Ingredient:

● Use this ingredient:

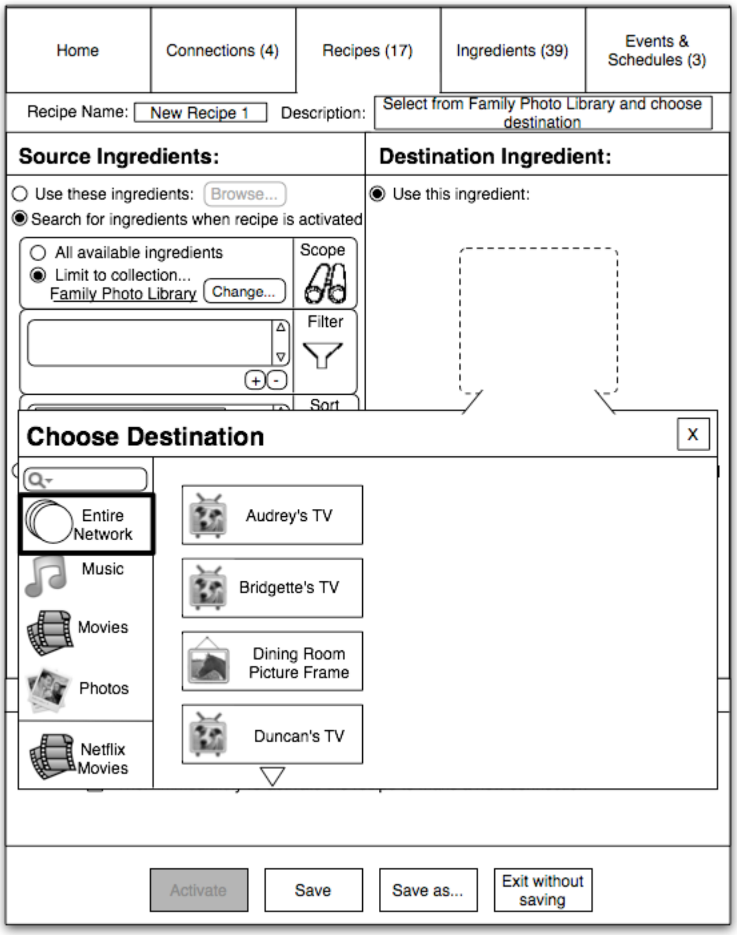
○ Search for ingredients when recipe is activated

Connection Options:

□ Stop connection after 5 minutes

□ Then immediately re-activate the recipe to make a new connection

Figure 5-19: If the user elects to "Search for ingredients when the recipe is activated," they are presented with a set of options that allow them to define the search criteria for the selected slot. In this example, the user is defining criteria for source components.



Recipe Name: New Recipe 1

Description: Select from Family Photo Library and choose destination

○ Use these ingredients:

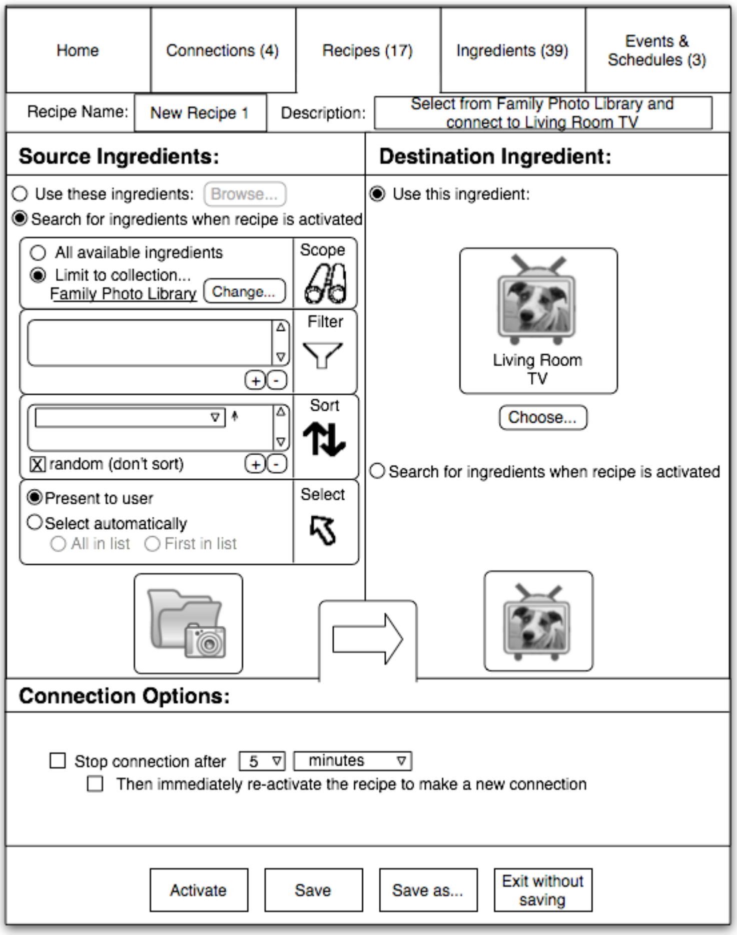
● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection... Family Photo Library

● Use this ingredient:

Figure 5-20: The user can also elect to specify one or more specific components for the source or destination slot. In this example, the user is browsing for a component to use as the destination of this template's connection.



Recipe Name: New Recipe 1
Description: Select from Family Photo Library and connect to Living Room TV

Source Ingredients:

○ Use these ingredients: Browse...
● Search for ingredients when recipe is activated
○ All available ingredients
● Limit to collection...
Family Photo Library Change...
☒ random (don't sort)
● Present to user
○ Select automatically
○ All in list ○ First in list

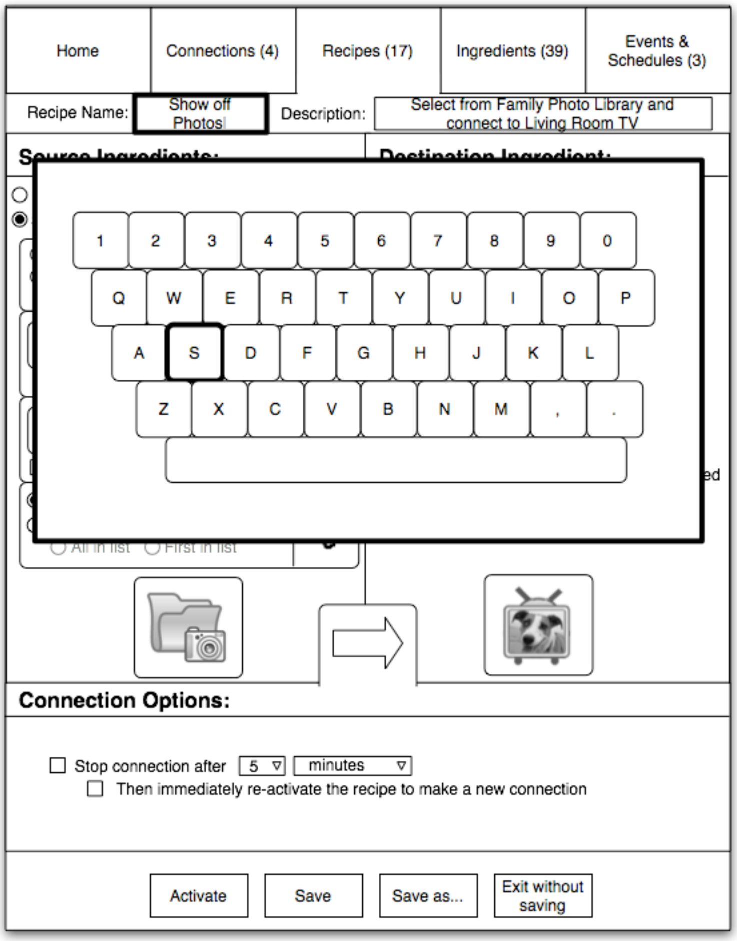
Destination Ingredient:

● Use this ingredient:
Living Room TV
Choose...
○ Search for ingredients when recipe is activated

Connection Options:

□ Stop connection after 5 minutes
□ Then immediately re-activate the recipe to make a new connection

Figure 5-21: After choosing a destination component, the recipe is shown as ready to activate, as evidenced by the now-enabled "Activate" button and the arrow with a solid border.



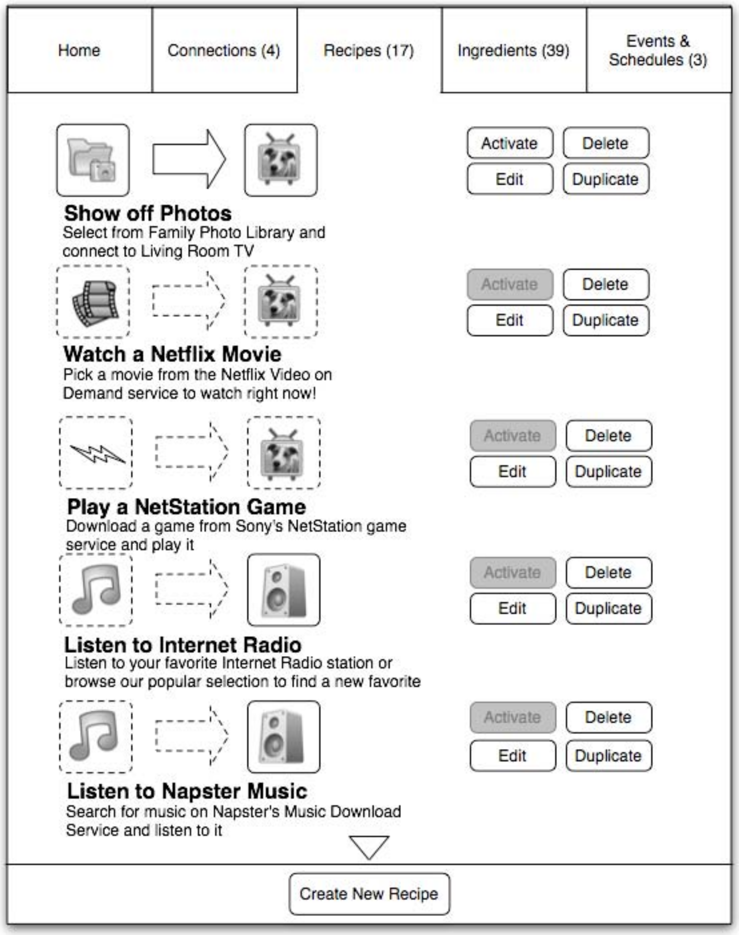
Recipe Name: Show off Photos

Description: Select from Family Photo Library and connect to Living Room TV

□ Stop connection after 5 ▽ minutes ▽

□ Then immediately re-activate the recipe to make a new connection

Figure 5-22: The user can give the recipe a name by tapping in the “Recipe Name” textbox and using the soft keyboard to enter a name.



Show off Photos
Select from Family Photo Library and connect to Living Room TV

Watch a Netflix Movie
Pick a movie from the Netflix Video on Demand service to watch right now!

Play a NetStation Game
Download a game from Sony's NetStation game service and play it

Listen to Internet Radio
Listen to your favorite Internet Radio station or browse our popular selection to find a new favorite

Listen to Napster Music
Search for music on Napster's Music Download Service and listen to it

Figure 5-23: After completing configuration of the new recipe, the user returns to the list of recipes and sees the newly created recipe at the top.

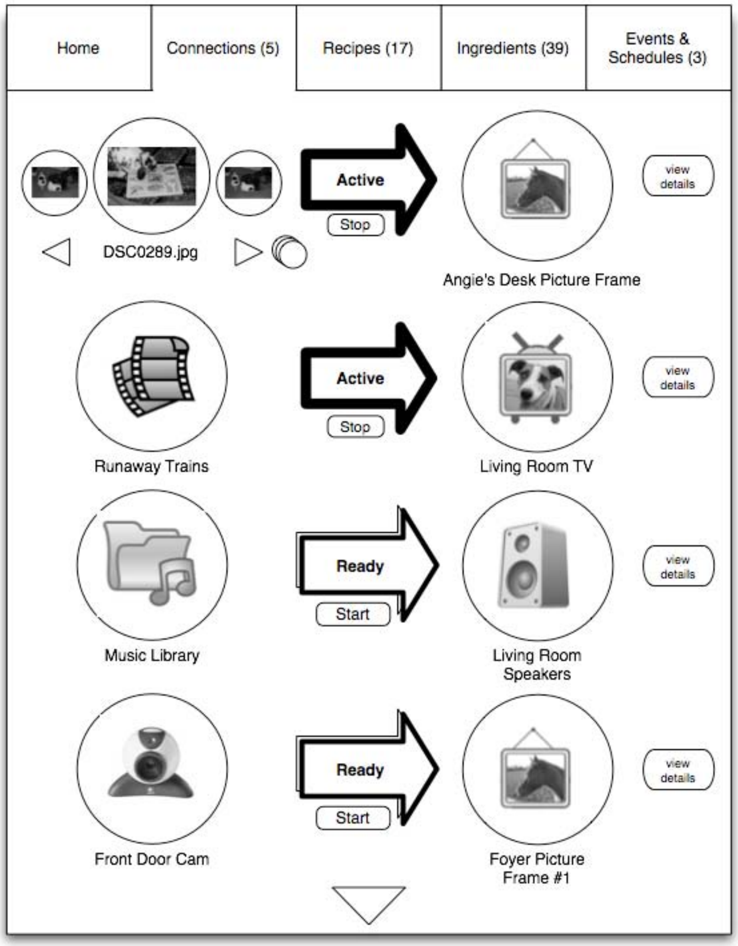


Figure 5-24: At any point, the user can tap the "Connections" tab in order to view the list of current connections. This list allows the user to monitor, control, and view details on any connection that OSCAR has created.

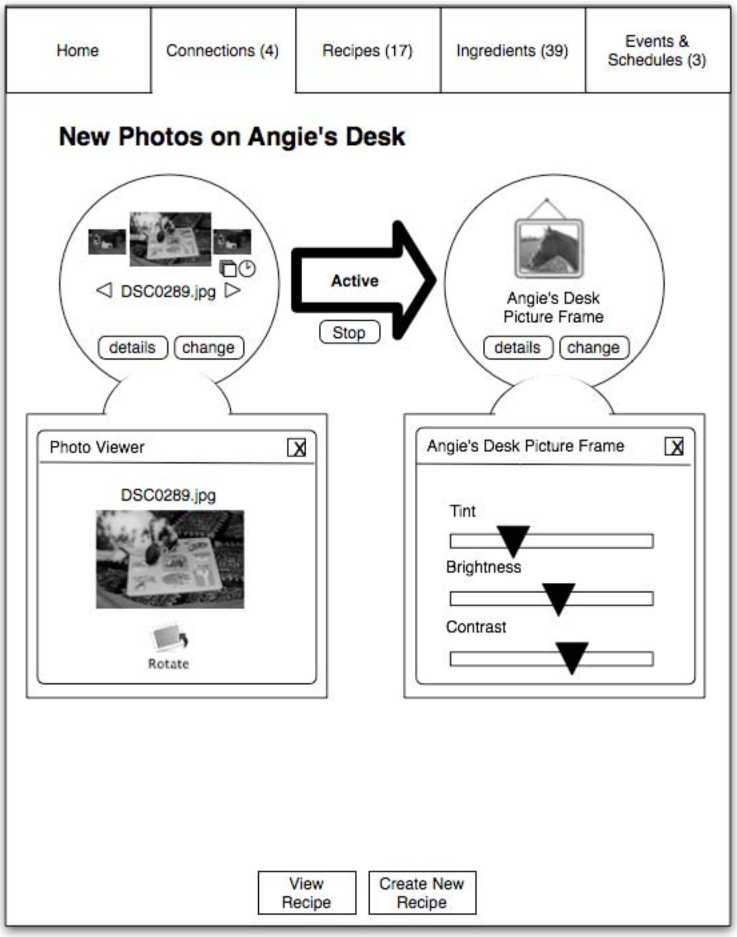


Figure 5-25: When viewing details on a connection, the user sees the currently connected components and the current status of the connection (Active, Ready, Stopped, Failed, Unavailable). In addition this screen provides controls to stop and start the connection and control the components involved in the connection.

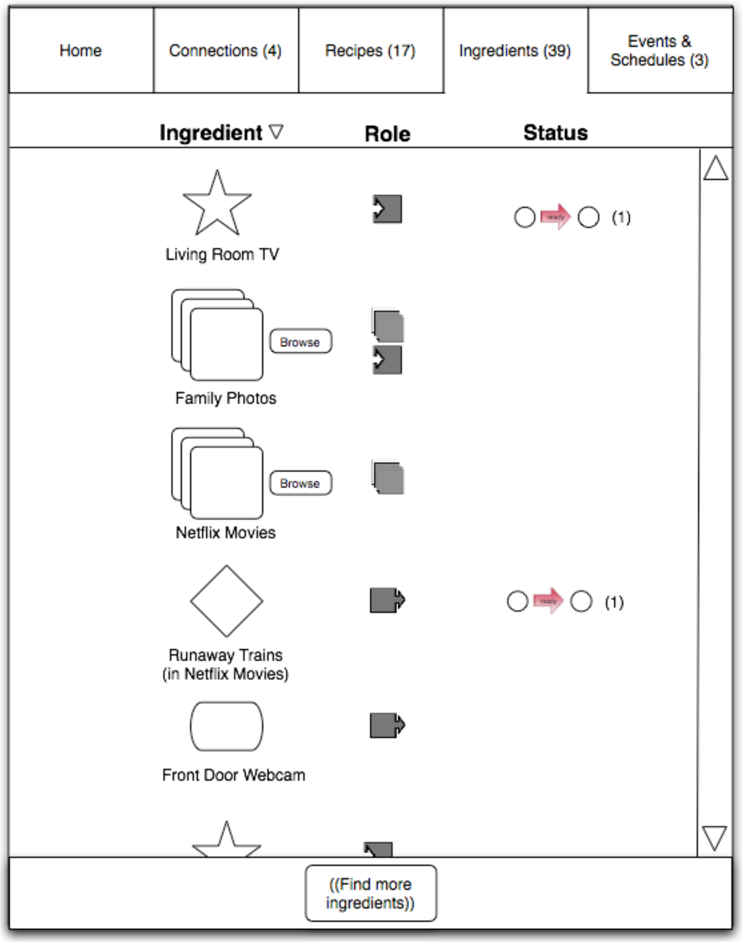
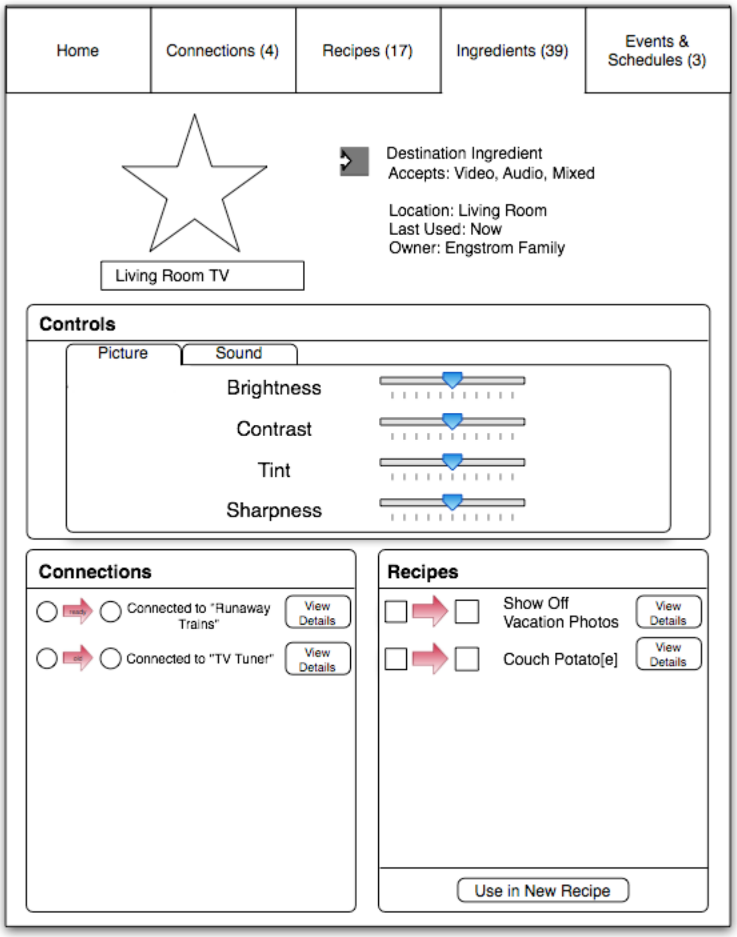


Figure 5-26: The "Ingredients" tab provides access to an always-available list of the components (devices and media) that have been discovered by OSCAR. The icons in the "role" column express the device/media source's role(s) in Obje data transfer and aggregation, and the graphics in the rightmost column are intended to show the number of connections in which the ingredient is involved.



Living Room TV

Destination Ingredient
Accepts: Video, Audio, Mixed

Location: Living Room
Last Used: Now
Owner: Engstrom Family

Connected to "Runaway Trains"

Connected to "TV Tuner"

Show Off Vacation Photos

Couch Potato[e]

Figure 5-27: The Ingredient Details screen provides access to the controls for the selected device, as well as information about which connections and recipes the devices is a part of.

5.2.5.3 Second Expert Evaluation

The medium fidelity walkthrough prototype was presented to three of the four experts that reviewed the first prototype (the fourth expert was unable to commit the time). I received feedback from the expert panel on a variety of issues. The overall feedback was positive, and the panel felt that the most significant issues identified in the first iteration had been addressed. There was universal praise for the new "recipe" metaphor and for the separation of recipes and connections into separate sections of the UI, though there were still some specific problems with clarifying the different information and capabilities associated with each of the two distinct concepts: a recipe (a data structure that describes how a connection or set of connections should be made) and a connection (an active session involving the transfer of data between components).

Support for browsing was considered to be greatly improved, though still amenable to further improvements, especially when browsing within media collections.

By and large the feedback was of a detailed nature—offering suggestions to improve the navigation options or language used in the UI. These improvements were incorporated into the next version of the prototype: the OSCAR1 Interactive Prototype.

5.3 OSCAR 1: The First Interactive Prototype

After the second, medium-fidelity prototype and expert evaluation, work began on the first interactive version of the OSCAR prototype. OSCAR1 was developed to run on the TabletKiosk Sahara Tablet PC shown in Figure 5-1. This device has a 12.1" LCD screen that is operated via touch input and runs Windows XP Tablet PC Edition. OSCAR1 was designed to use the entire screen and to hide all of the OS widgets and controls so that users were largely unaware that they were interacting with a computer application running on a conventional PC. Our goal was to simulate a new, unfamiliar device with new, unfamiliar capabilities.

5.3.1 The OSCAR Testbed

A dedicated room at PARC served as a site for development, experimentation, and ultimately user evaluations. The room featured a large comfortable couch positioned in the "living room" area, a refrigerator in a corner designated as the "kitchen" area, and another area near the entryway designated as the "front door." We deployed a set of media services and devices in and around this room, each of which were implemented as Obje components. In all there were sixteen Obje components of nine different types (3 speakers, 2 TVs, 2 picture frames, 2 generic screens, 3 microphones, a webcam, a music library, a photo library, and a TV tuner) hosted on 5 networked computers in the room and throughout our lab. Figure 5-28 shows a schematic of the testbed room.

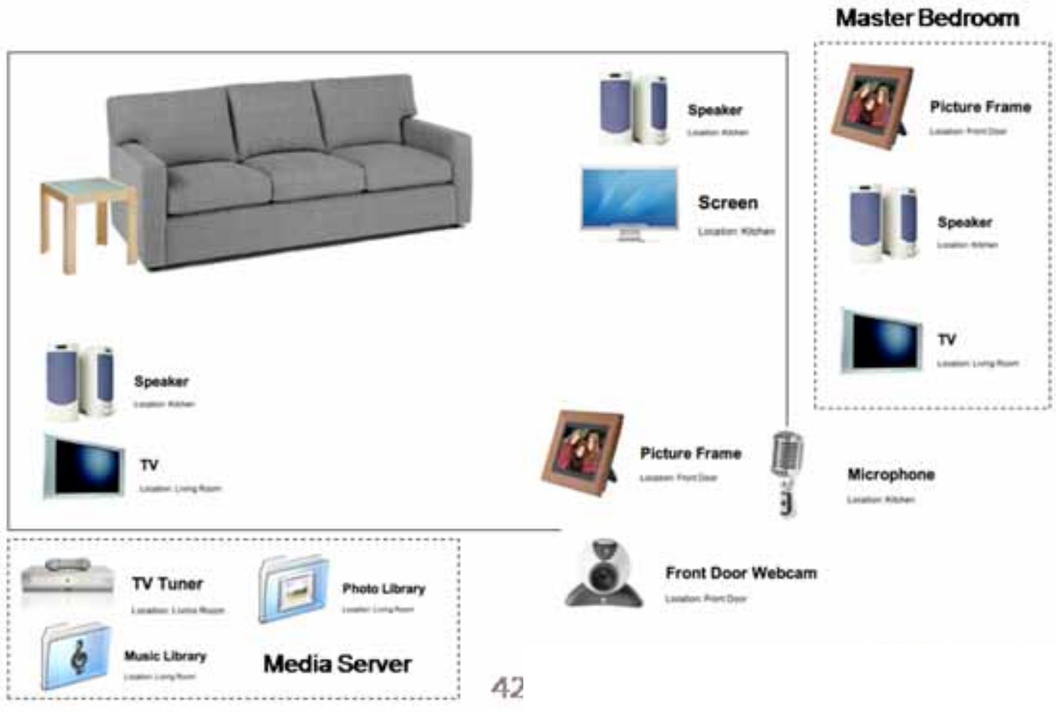


Figure 5-28: A schematic of the room that was used as a testbed for the OSCAR development and user studies.

5.3.2 The OSCAR1 User Interface

OSCAR1 did not differ much from the OSCAR medium-fidelity prototype in its functionality or its basic user interface. The key differences between OSCAR1 and its predecessor include:

- Three global tabs instead of five and the replacement of the never-developed “home” screen with starting the user at the “Recipes” screen
- A significantly redesigned recipe details screen that includes:
    - Abandonment of the explicit query building UI shown in Figure 5-19 and its replacement with the option-based UI shown in Figure 5-30.
    - Closer integration of static ingredient and dynamic ingredient (criteria-based) slot definitions.
    - Refactoring of playlist/slideshow rules into the source slot (see Figure 5-30) as opposed to presenting them as connection-level attributes (Figure 5-19).
- A unified color scheme to reinforce the different sections of the interface represented by the global tabs.

In the OSCAR1 prototype pictured in Figure 5-29 through Figure 5-35, the user starts with the *Recipes* screen shown in Figure 5-29. From the *Recipes* screen, users browse a list of available recipes and “prepare” them to activate connections among media and devices. From this screen, they can also create a new recipe (by tapping “New Recipe”) or edit an existing recipe (by tapping “Details”). In either case, they would be directed to the *Recipe Details* screen for the selected/created recipe. (Figure 5-30). The choices the user makes in the details screen for any given recipe will affect what happens when the recipe is "prepared." The recipe details will define candidate components for both the source and destination slots as well as selection options dictating how the candidate list would be used to populate the slot when the recipe was prepared. In the example shown in Figure 5-30, for example, the user is editing a recipe whose name is "Who's at the door?" It is configured to always use the "Front Door Webcam" as the source component, as indicated by the fact that the "Single Ingredient" tab is selected in the source slot panel, the "Front Door Webcam" component is selected as the single ingredient, and the option "Use them all" is chosen (though this last option is redundant since only one, non-Aggregate component has been chosen). This recipe is further configured to prompt the recipe's user to select among the "Picture Frame in Front Door" and "TV in Living Room" when the recipe is prepared. This is indicated by the fact that the "Ingredient List" tab is chosen, the two aforementioned components are in the list, and the option "Show them to me and I will choose" is selected below the list.

After preparing a recipe, which can be done from either the Recipes or Recipe Details screens, an active connection is created and the user is directed to the Connection Detail screen. In the example shown in Figure 5-31, an active connection between a playlist consisting of Led Zeppelin songs and the Living Room speakers is being depicted. The playlist contents, currently playing song, connection status, and controllers for both the audio renderer and speakers are all visible on this screen. Recall that controller user interfaces such as these are provided by the components themselves through the Obje Framework's mobile code delivery mechanism, and they are rendered by OSCAR with no advance knowledge of their capabilities [111].

If the user wishes to select a different source (in this case, select different music to play), they would simply tap the button labeled "Change" underneath the source component's icon. Doing so results in the screen shown in Figure 5-32, in which a browse dialog is displayed to allow the user to select an alternative source. Note that if the user had prepared a recipe that was set to "Show me and I will choose," a browse dialog such as the one shown in Figure 5-32 would have been shown immediately after the recipe was prepared, and before any connection was made active. Multiple connections can be monitored and controlled simultaneously using the *Connections* screen (Figure 5-33).

An alternative way to control devices and create new recipes is to start with the Ingredients screen shown in Figure 5-34. From here, the user can view all of the components that have been discovered on the network and tap on their icons to view their details (Figure 5-35). The details include the control (or "admin") user interface that is downloaded automatically and displayed to the user. The details also show any connections or recipes that involve this component. In addition to allowing the user to manipulate the control UI, the user can choose to create a new recipe by tapping “Use in New Recipe.”

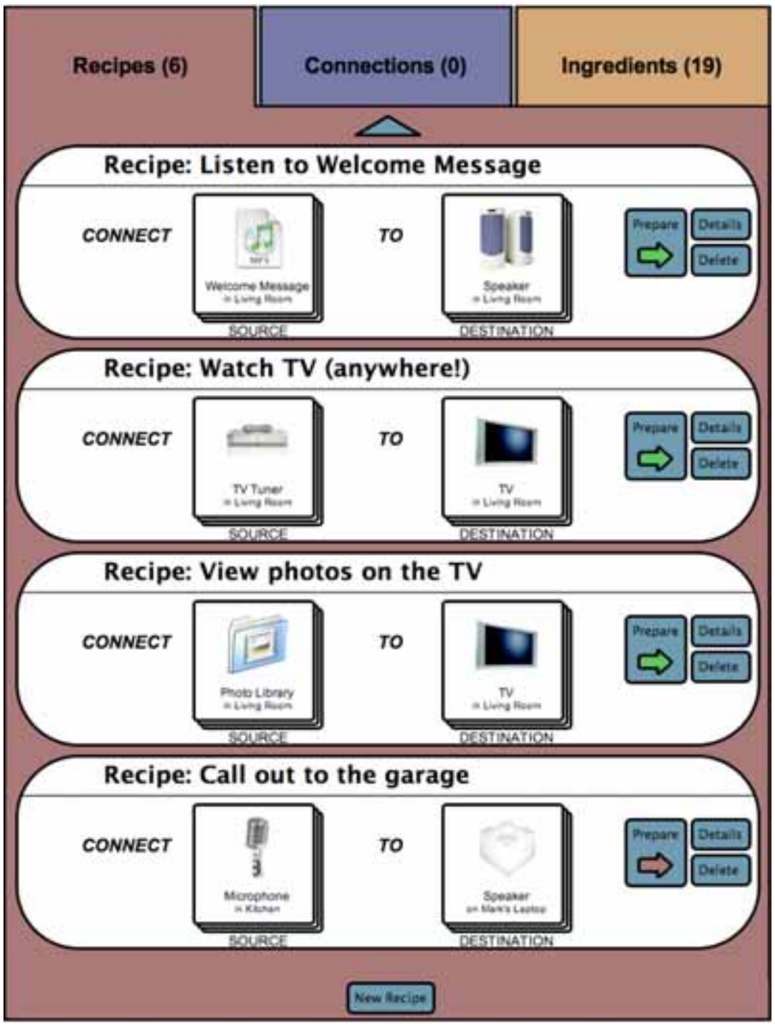
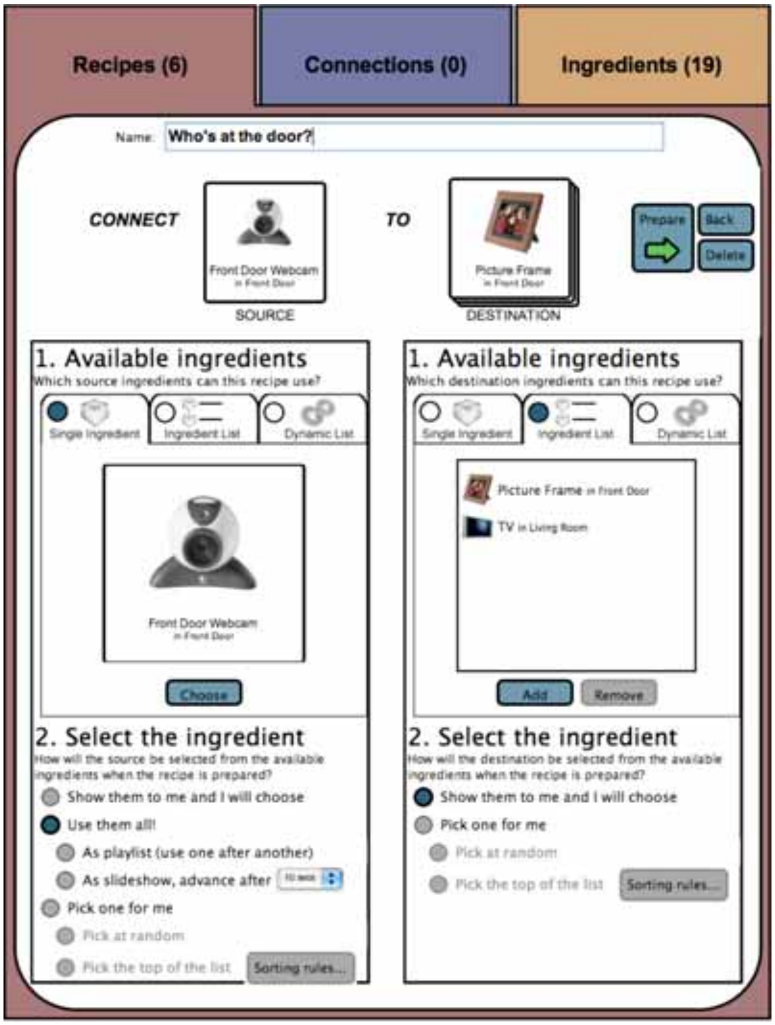


Figure 5-29: The OSCAR1 list of recipes. This is the first screen that a user sees when interacting with OSCAR1. From this screen, the user can “prepare,” delete, or view details/edit a recipe. In addition, from this or any screen the user can tap the globally available tabs at the top of the screen to jump to a different part of the UI.



Name: Who's at the door?

1. Available ingredients

Which source ingredients can this recipe use?

● Single Ingredient ○ Ingredient List ○ Dynamic List

Front Door Webcam in Front Door

2. Select the ingredient

How will the source be selected from the available ingredients when the recipe is prepared?

○ Show them to me and I will choose

● Use them all!

○ As playlist (use one after another)

○ As slideshow, advance after 10 secs

○ Pick one for me

○ Pick at random

○ Pick the top of the list

1. Available ingredients

Which destination ingredients can this recipe use?

○ Single Ingredient ● Ingredient List ○ Dynamic List

Picture Frame in Front Door

TV in Living Room

2. Select the ingredient

How will the destination be selected from the available ingredients when the recipe is prepared?

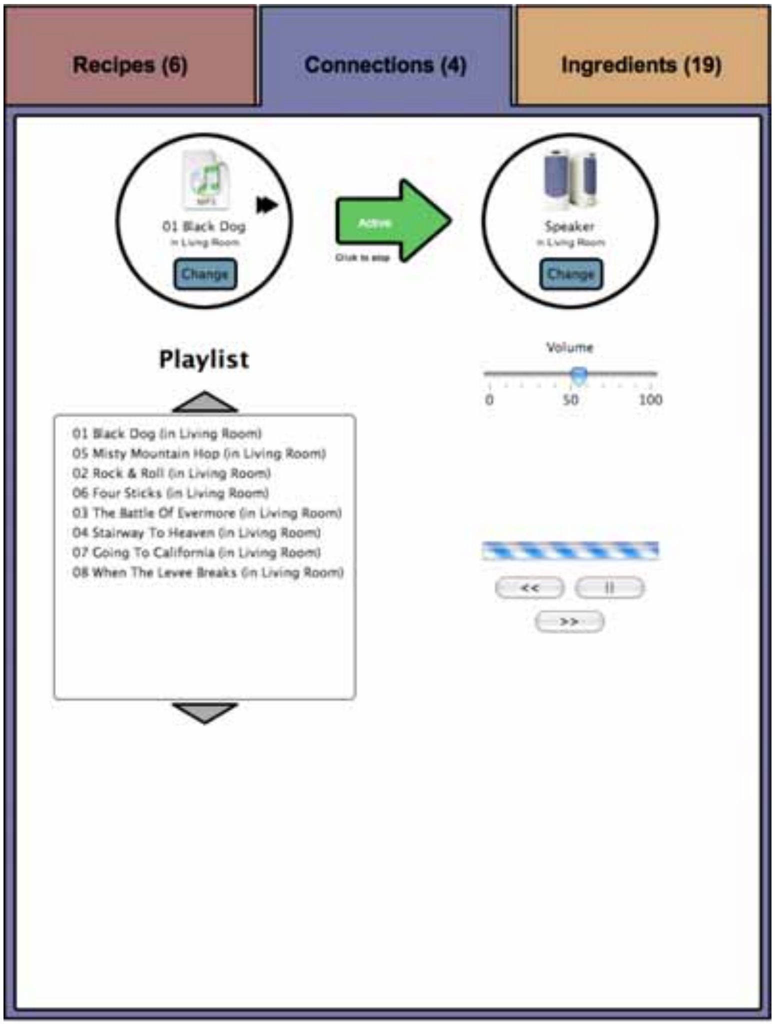
● Show them to me and I will choose

○ Pick one for me

○ Pick at random

○ Pick the top of the list

Figure 5-30: The user can edit a recipe by defining specific components to be used in a connection, as shown here, or by defining "Dynamic List" criteria. In addition, for situations where more than one component can populate a source or destination slot, the user can define selection rules (shown under the heading "2. Select the ingredient" that will determine how the component will be selected when the recipe is prepared.



01 Black Dog (in Living Room)
05 Misty Mountain Hop (in Living Room)
02 Rock & Roll (in Living Room)
06 Four Sticks (in Living Room)
03 The Battle Of Evermore (in Living Room)
04 Stairway To Heaven (in Living Room)
07 Going To California (in Living Room)
08 When The Levee Breaks (in Living Room)

Figure 5-31: After preparing a recipe, a connection is created. In some cases, depending on the selection rules defined by the user and the available components at the time of preparation, the result is an active connection like the one shown here. In this case, a playlist of Led Zeppelin songs is being streamed to the living room speakers.

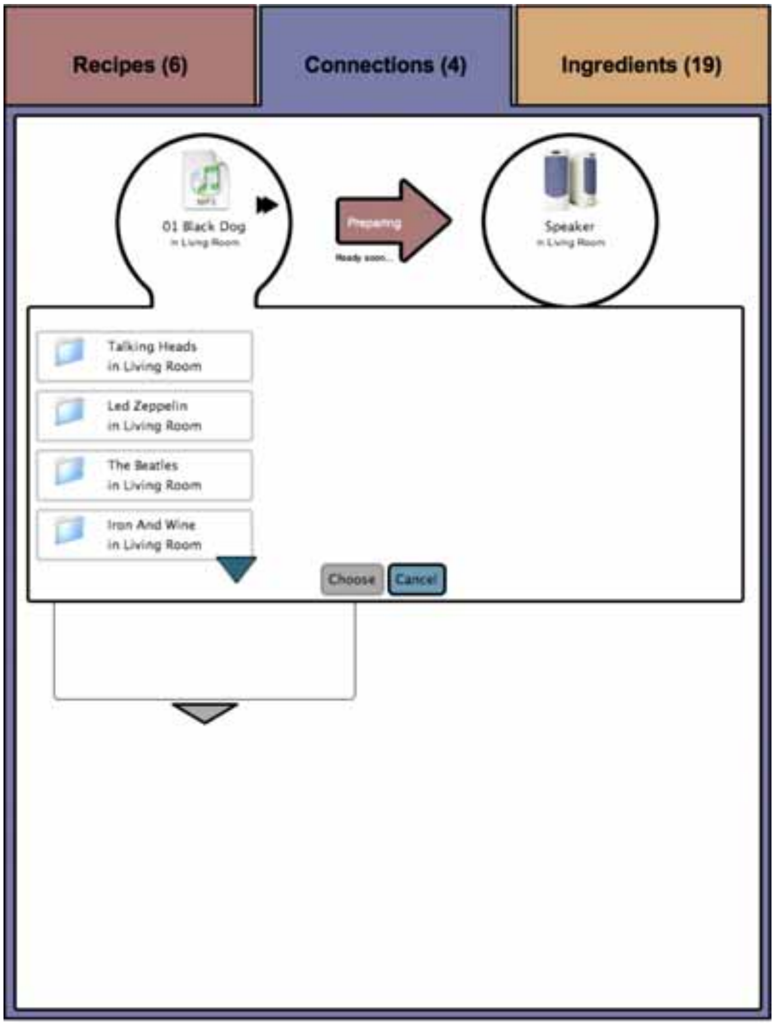


Figure 5-32: At any point during the life of a connection, the user can elect to change the source or destination component being used. By tapping the "Change" button on the screen shown in Figure 5-31, the user is presented with a dialog that allows them to browse for a replacement component.

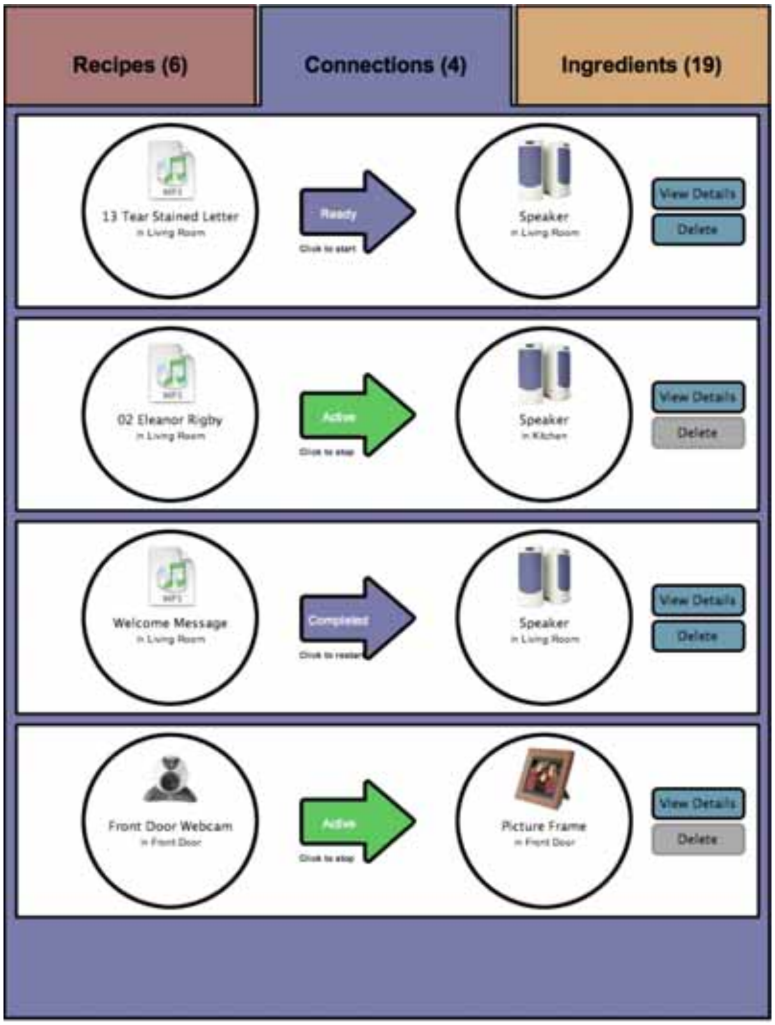
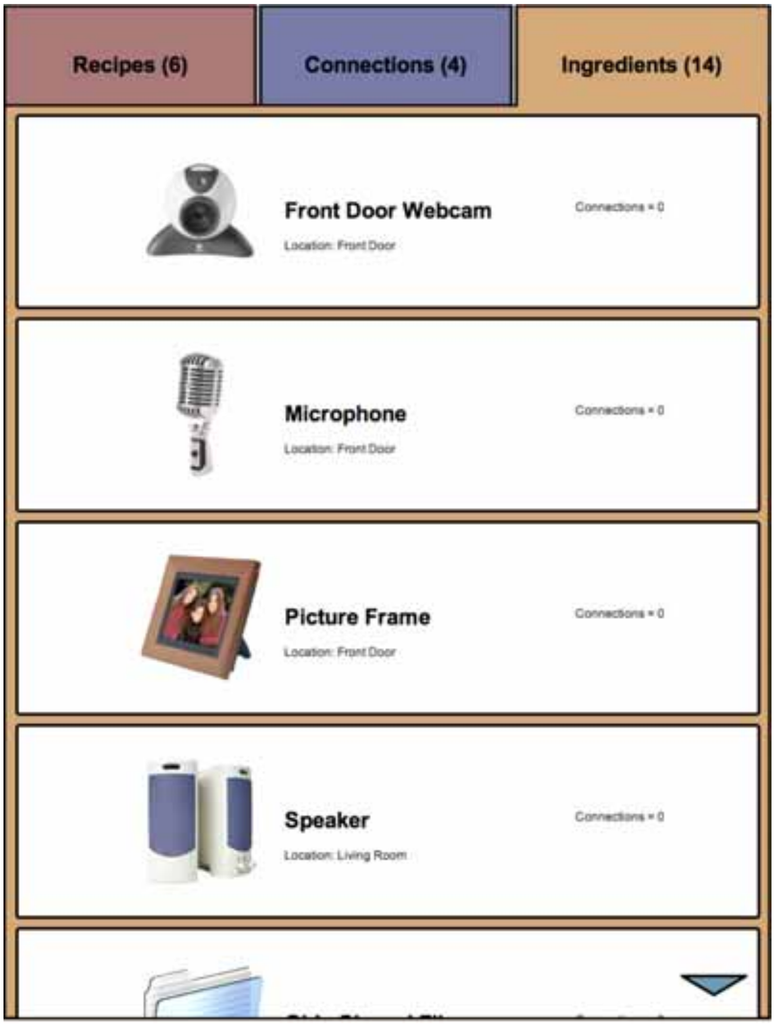


Figure 5-33: The user can monitor and control multiple connections at the same time via the Connections List screen. This screen is always accessible via the "Connections" tab.



Front Door Webcam
Location: Front Door
Connections = 0

Microphone
Location: Front Door
Connections = 0

Picture Frame
Location: Front Door
Connections = 0

Speaker
Location: Living Room
Connections = 0

Figure 5-34: The list of available devices and media is also always available via the "Ingredients" tab. From here, the user can view details on any component that OSCAR has discovered on the network.

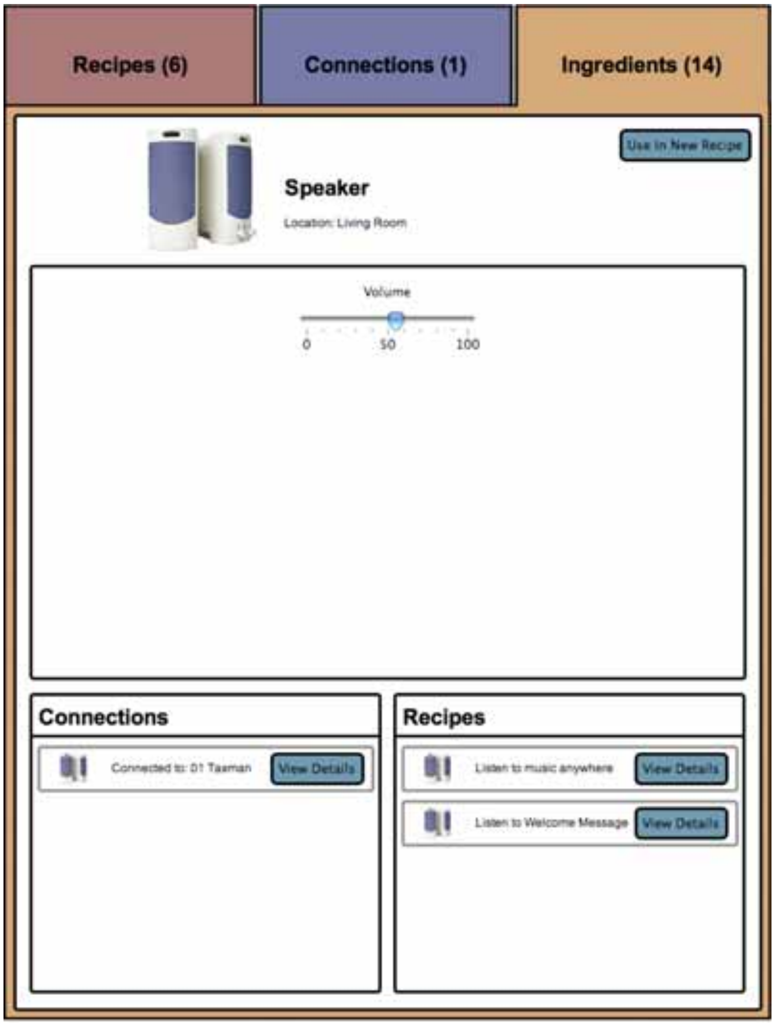


Figure 5-35: Via the Ingredient Detail screen, the user can access controls specific to the selected device, view and access any connections or recipes in which the device is involved, and create a new recipe that uses this device.

The walkthrough just presented demonstrates most of the key functionality available in OSCAR1. We have seen how a user can

- View and select from the list of recipes
- Browse the list of devices and media that are available on the network
- Incorporate those devices and media into templates and connections
- Edit a template to define rules regarding how selected devices and media are connected at template activation time
- Activate a template and view the resulting connection(s)
- Gain access to controls for stopping and starting connections
- Gain access to device- and media-specific controls for an active connection

The OSCAR paper prototype was designed to afford a handful of additional capabilities that were not highlighted by the foregoing walkthrough. Additional options include:

- Configuring a template to automatically select a new item from the list of sources at a particular time interval (slideshow mode)
- Configuring a template to automatically select a new item from the list of sources upon completion of the current source's connection (playlist mode)
- Defining custom criteria to generate the list of source or destination components

Note that while these latter functions are still present in OSCAR1, as they were in earlier versions of the prototype, through the development of the usage scenarios, user study plans, and personas, they had come to be seen as less important than the core functionality shown in the foregoing walkthrough. That is to say, it had become apparent that these functions would be of only marginal utility to many users, and that it would be most productive to focus the bulk of development and user testing on the core functions of discovering and controlling devices, making and controlling connections, and creating reusable compositions of specific, known devices.

5.4 Summary: Towards OSCAR2

OSCAR1 was the culmination of many months' design, evaluation, and implementation. Once OSCAR1 had reached an acceptable level of completeness and stability, I designed and conducted a user study to evaluate its effectiveness against the goals outlined at the beginning of this chapter and to contribute to OSCAR's continued improvement. The next chapter covers the OSCAR1 user study and results, the changes and improvements suggested by those results that resulted in OSCAR2, the second user study that examined OSCAR2, and the results of that second study.

6 The OSCAR User Study

As described in Chapter 5, OSCAR was designed to explore the feasibility of end-user composition and the design issues with delivering composition tools to end-users. As such, it also provides a window into the user experience goals of the Obje Framework, upon which OSCAR is based. In addition, a major goal in the design of OSCAR was to provide a high-quality application to support users in the discovery, control, and composition of media-related resources in the home.

To assess OSCAR's success along each of these dimensions, I designed and conducted a user study. The user study was conducted in two phases using two versions of the OSCAR prototype. The second phase of the study followed the first phase by approximately eight weeks, during which time we made numerous changes to OSCAR based on the results of the first phase. None of the changes were planned in advance save one: as discussed earlier, there were significant questions about the most efficacious "world view" to present to users: should OSCAR be presented as a device/media-oriented application that also supports re-usable configurations, or should it be presented as an activity-oriented application that provides access to devices and media through the construct of "recipes" or some other framing of activity-oriented custom configurations? To explore this question, it was determined in advance that OSCAR1 would adopt the “Activity-oriented” world view, and that OSCAR2 would adopt the “Devices and Media-oriented” world view. No other differences between the two versions were decided in advance of the first study phase, though as will become apparent later in this chapter, it was necessary to make several other changes to improve the overall usability of the system.

In all, the study included 18 participants recruited from the surrounding community, none of whom were professional programmers or system administrators. Nine participants were assigned to the first phase and the other nine were assigned to the second. No users participated in both phases.

In this chapter I will describe the user study design, the results of the first phase of the study, the changes that were made to the prototype before the second study phase, and the results of the second phase. In addition I will discuss the results of the in-depth interview that was conducted with all eighteen participants and the implications of these results for future design.

6.1 User Study Design

As described in Chapter 5, a room at PARC was dedicated to the design and evaluation of OSCAR as a testbed and study site (see Figure 6-1). The space simulated a home with a large comfortable couch positioned in the “living room” area, a refrigerator in a corner designated as the “kitchen” area, and another area near the entryway designated as the “front door.” We deployed a set of media services and devices in and around this room, each of which were implemented as Obje components. In all there were 16 components of 9 different types (3 speakers, 2 TVs, 2 picture frames, 2 generic display screens, 3 microphones, a webcam, a music library, a photo library, and a TV tuner) hosted on 5 networked computers throughout our lab.

All user study sessions were held in the OSCAR room and lasted approximately 90 minutes (see Figure 6-2). The study participants were screened such that they were not professional system administrators or programmers. We had fairly even distributions of gender (10 male, 8 female), age (20-55, average=36.5), and education level (“some college” through “graduate degree”). The complete demographics are included in Appendix F. I was present for every session along with my colleague Ame Elliott, who assisted by acting as “host” for each session (i.e., handling most of the interactions with the participant) and contributing to the observations of participants’ performance and reactions. In addition to extensive notes, each session was videotaped with two cameras (see Figure 6-3) and screen capture software [146] recorded user actions. After each session, the two research team members (Ame and myself) independently scored each participant on a number of variables, including the amount of help required to perform each task, the correctness of the solution to each task, overall comprehension of the system, and the participant’s experience with computer technology and home networking (based on responses to interview questions). In a post-session debriefing meeting, the two team members discussed their independent scores and reached a consensus. Since the scoring criteria for factors such as amount of help required and for solution correctness were being developed as the study progressed, we also went back after all the participants had finished and reviewed the video to ensure that all participants' task performance had been graded consistently. A handful of scores were corrected during this process, but most were left intact.

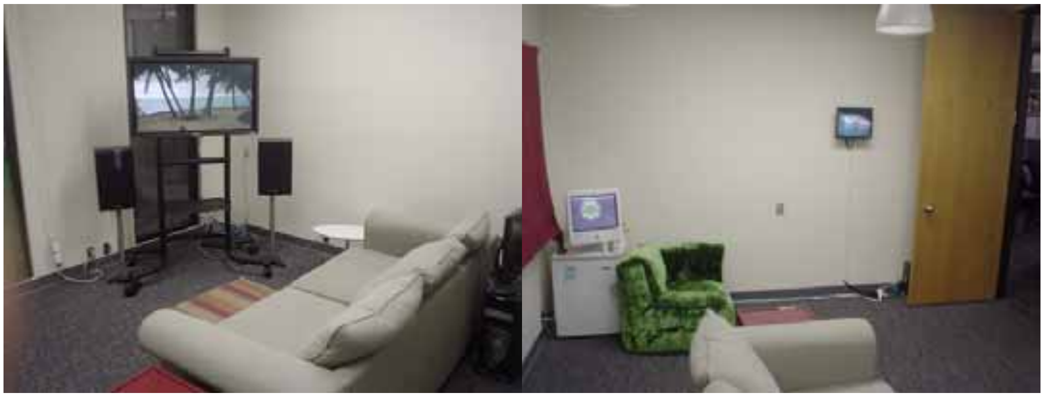


Figure 6-1: The study setup used for the OSCAR user study. These images show the testbed room from two angles, showing the “living room” (left) and “kitchen”/ “front door” (right).

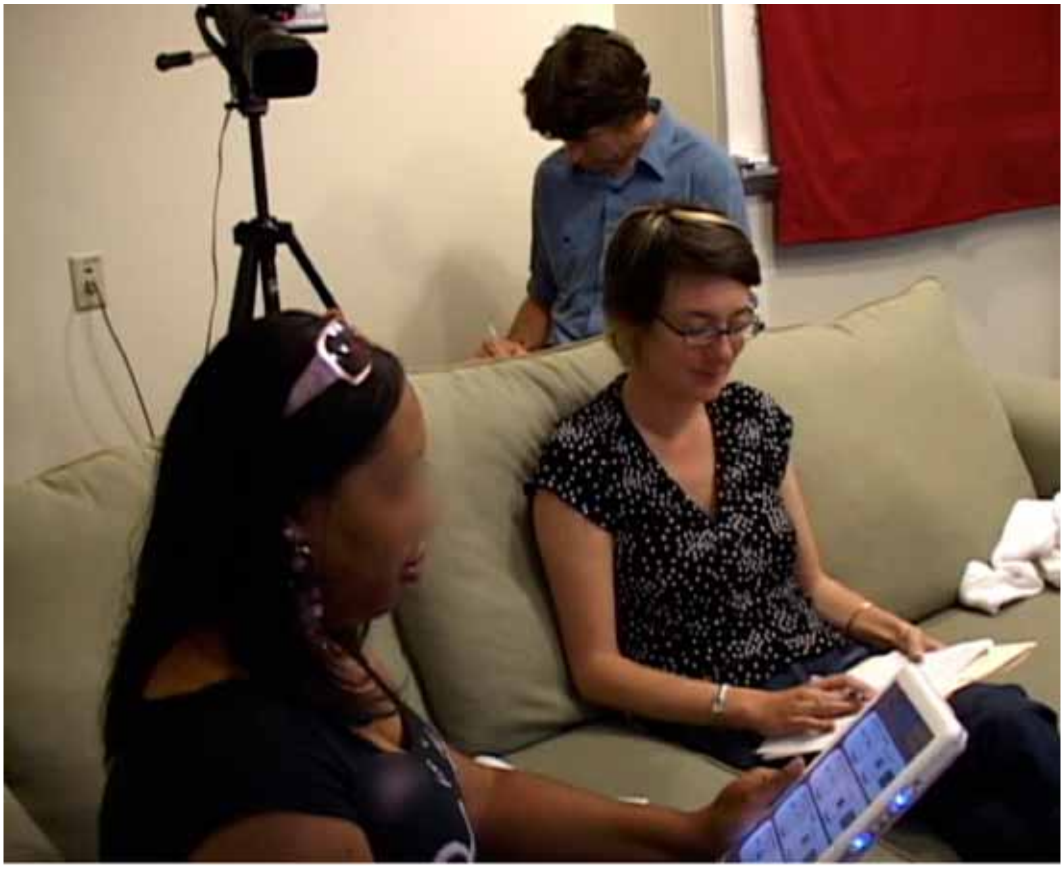


Figure 6-2: The test setup used for the OSCAR user tests. One of the research team members (on the couch, to the right) handled all the communication with the participant (on the couch, to the left), except for the interview, which was conducted by both research team members. The second team member (behind the couch) was responsible for running the video cameras and setting up and maintaining OSCAR and all of the Obje components before and during the sessions.

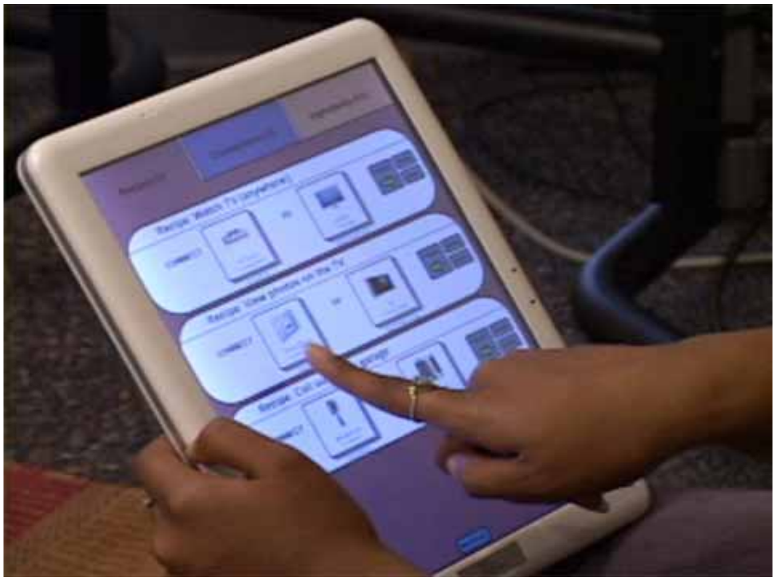


Figure 6-3: An example of one of the OSCAR1 participants interacting with OSCAR during the session. This shot represents the view captured by the "over-the-shoulder" camera.

6.2 User Study Sessions

Each user study session consisted of a number of different activities, including introduction and scene setting, tasks, questionnaire, screen-by-screen usability assessment, and follow-up interview. I will describe each activity in the following subsections.

6.2.1 *Introduction and scene setting*

We asked each participant to imagine that they were sitting in their own homes of the near future. We gave them a brief orientation to "their" house, pointing out the various rooms as well as the wirelessly connected devices, and media libraries. As we handed them the tablet running OSCAR, we asked them to imagine that they had just purchased a new device from a consumer electronics store, and that what they knew about this device was that it would allow them to connect and control the other devices in their homes. We simulated an "out-of-the-box" experience, in which a consumer acquires a new device and sets it up with no help or instructions. Participants received no training at all, and were not given any information about any of the functions or concepts in the interface, such the meaning of the term "recipe."

6.2.2 Tasks

We then presented the participant with a set of four tasks, presented one at a time. As they carried out the task, participants were encouraged to think aloud as they worked. The tasks were as follows:

Task 1) Play the welcome message on the Living Room Speakers.

Task 2) Show a picture from your recent trip to Australia on the Living Room TV. Then show a different photo from the same trip.

Task 3a) Someone has just rung the doorbell. See who it is by displaying the image from the webcam outside the front door on the picture frame just inside the front door.

Task 3b) Now make it so that you or someone you live with can easily do this again, without having to redo it from scratch. Give it a name so that you will remember what this does.

Task 3c) Now make it so that when the doorbell rings, you can decide to either show the image from the webcam on the picture frame OR the living room TV.

Task 4a) Someone you live with is more motivated to clean when they are listening to music. Make a recipe that will allow them to play all of the songs by the Beatles on the Kitchen speakers.

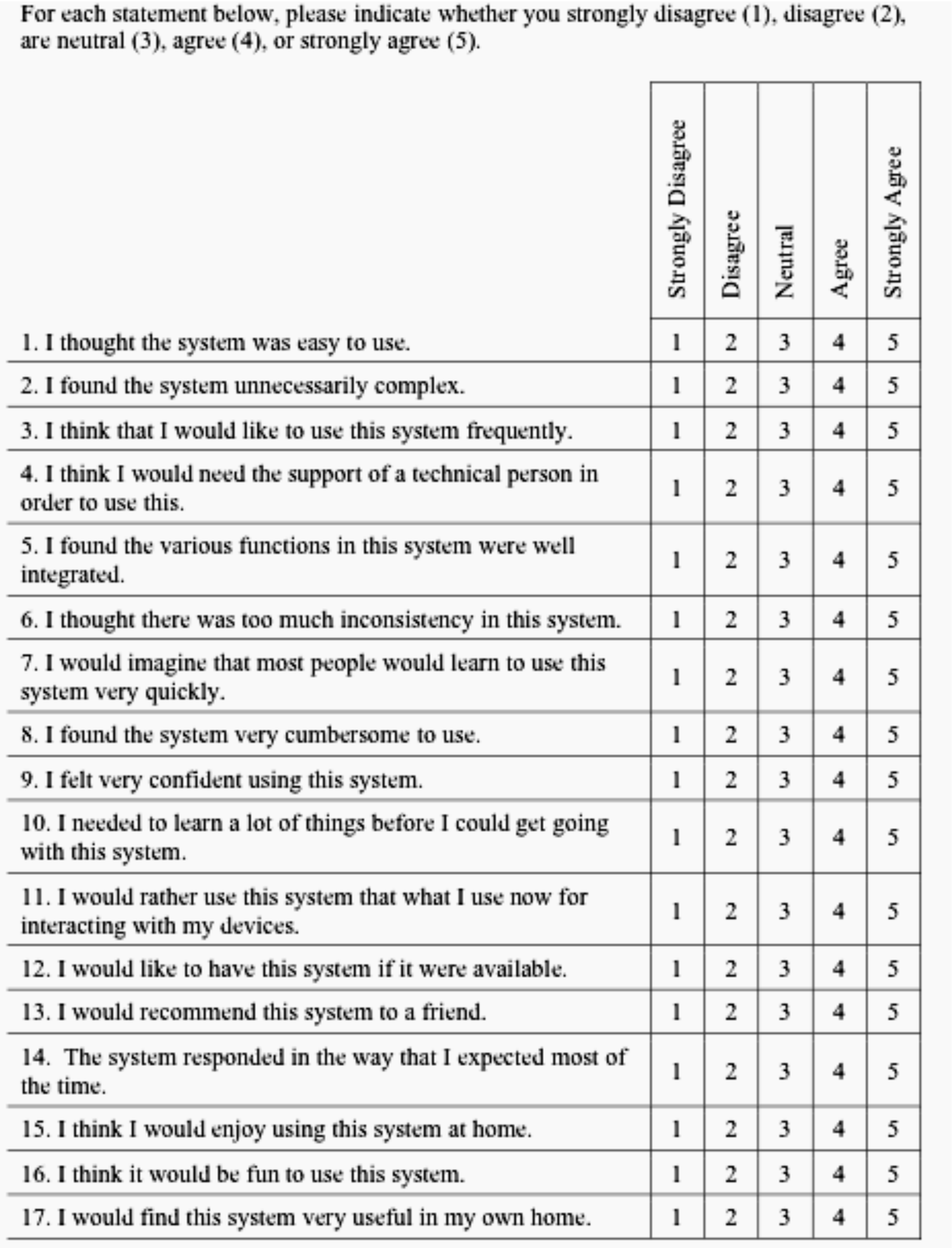
Task 4b) The person you made this for asks you to change this so that they can listen to other music, too, and do it any room of the house. Change the recipe so that they can pick the artist and the speakers when they activate the recipe.

Tasks 1 and 2 could be accomplished using recipes that were pre-loaded into OSCAR. In the course of Task 3 users needed to discover how to create a new recipe, and in Task 4 they created a different recipe, again from scratch.

6.2.3 Questionnaire

After completion of the four tasks, we administered a questionnaire to each participant to capture their reaction to OSCAR. The first part of the questionnaire comprised the System Usability Scale (SUS) [16], and the second part consisted of statements intended to assess the participant's projected subjective preference for using OSCAR in their own home. The first ten questions in the questionnaire shown in Figure 6-4 are directly adapted from the SUS. The next seven, excepting question 14[10], are the subjective preference questions.

[10] Question 14 was an additional usability question that was later deemed redundant with the SUS questions, and its results are not reported in this document.



For each statement below, please indicate whether you strongly disagree (1), disagree (2), are neutral (3), agree (4), or strongly agree (5).

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I thought the system was easy to use.	1	2	3	4	5
2. I found the system unnecessarily complex.	1	2	3	4	5
3. I think that I would like to use this system frequently.	1	2	3	4	5
4. I think I would need the support of a technical person in order to use this.	1	2	3	4	5
5. I found the various functions in this system were well integrated.	1	2	3	4	5
6. I thought there was too much inconsistency in this system.	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly.	1	2	3	4	5
8. I found the system very cumbersome to use.	1	2	3	4	5
9. I felt very confident using this system.	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system.	1	2	3	4	5
11. I would rather use this system that what I use now for interacting with my devices.	1	2	3	4	5
12. I would like to have this system if it were available.	1	2	3	4	5
13. I would recommend this system to a friend.	1	2	3	4	5
14. The system responded in the way that I expected most of the time.	1	2	3	4	5
15. I think I would enjoy using this system at home.	1	2	3	4	5
16. I think it would be fun to use this system.	1	2	3	4	5
17. I would find this system very useful in my own home.	1	2	3	4	5

Figure 6-4: The subjective usability and preference questionnaire that was administered to OSCAR user study participants. Questions 1-10 were derived from Brooke's System Usability Scale instrument [16], and the latter questions were designed to measure participants' preference for OSCAR as compared to existing alternatives.

6.2.4 Screen-by-screen usability assessment

We then presented each user with a printout of each screen in the UI and asked them to indicate with stickers parts of the UI that they found particularly helpful or appealing and parts that they found particularly confusing or distasteful. They used smiley-face stickers to indicate positive aspects and "!" stickers to indicate problem areas. Examples are shown in Figure 6-9 through Figure 6-11, Figure 6-18, and Figure 6-19. We discussed each judgment with them to ensure that we understood the reasons for their assessment.

6.2.5 Interview

The final part of each session consisted of a 30-45 minute interview in which we asked them a number of questions aimed at understanding their current and envisioned uses of home media networking technology, and how they could incorporate OSCAR's capabilities in their homes. We also asked questions relating to the desirability of a possible future development path: the sharing of devices and media across homes and the sharing of compositions among users in different locations. The interview was structured around a number of set topics, though the discussions were allowed to range freely in order to allow participants to express their interests and reactions as naturally as possible. The interview script (included in Appendix E) included questions to cover the user's comprehension and impression of the system as well as the desirability of incorporating a system like OSCAR into their own homes and what they would use it for if it were available to them.

6.2.5.1 Comprehension and impression

The first set of questions was intended to calibrate the participant's understanding of the functionality made available in OSCAR and the terminology used to describe it. Also, I attempted to calibrate the participants' perception of OSCAR's usefulness by asking them to describe what kind of person they believed would get the greatest use from OSCAR. The following questions were asked:

- What is a recipe?
- What is a connection?
- What is an ingredient?
- How would you describe this to someone else?
- What is your opinion of the language used in this application, especially "Recipe"?
- Who is a system like this for?

6.2.5.2 Recipe/Setup usefulness and preference

The next set of questions were intended to get a more complete picture of what the participants believed they would use OSCAR for. A number of different approaches were taken to try to get this information, including asking open-ended questions and using constrained instruments to gain comparable data across participants. In terms of the latter, I first asked participants to rank the relative usefulness of a set of recipes that would be used entirely within their own homes using the form shown in Figure 6-5. These were reflective of the types of recipes that were created and used during the task portion of the study.

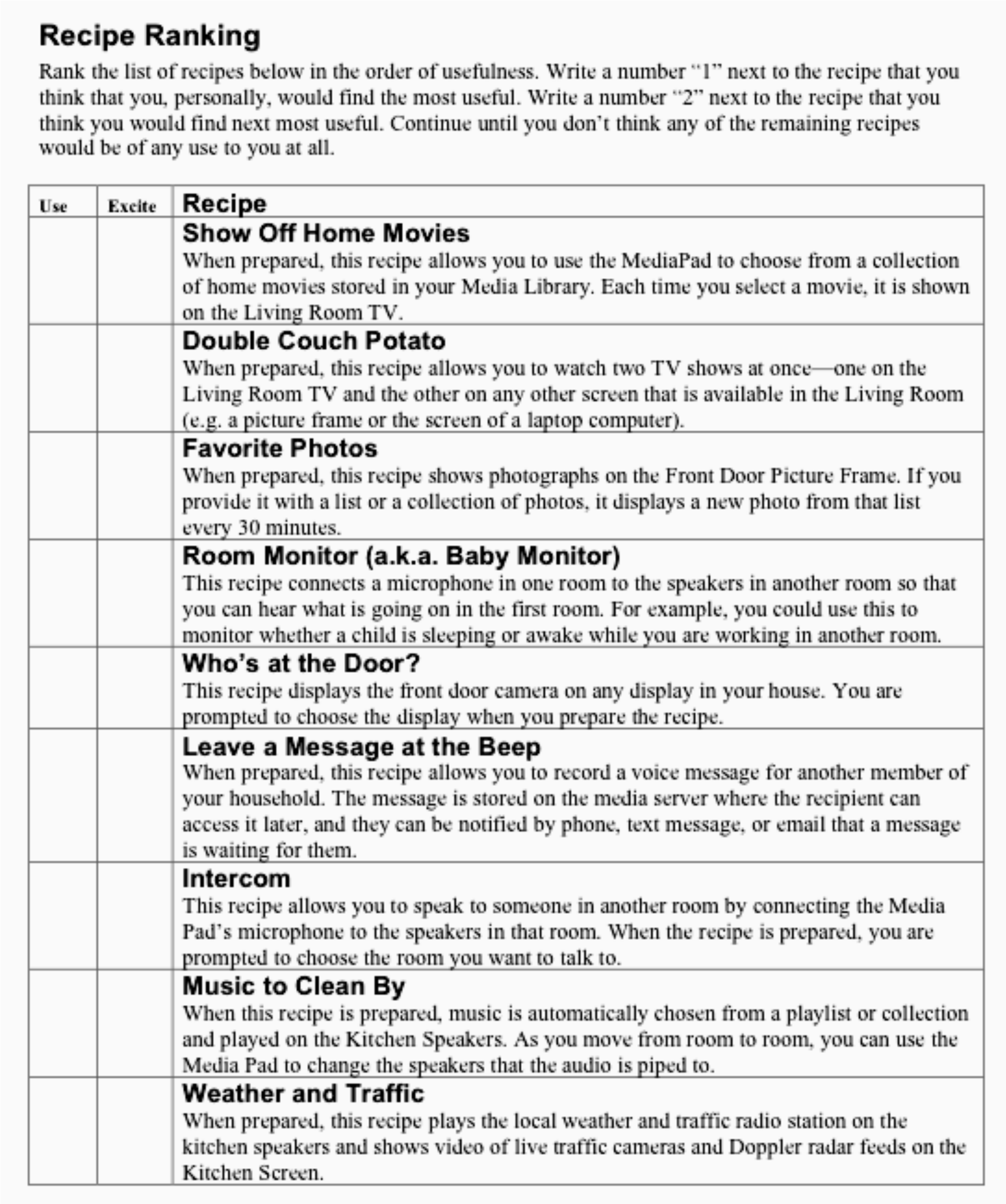
Following this, I asked participants to rank a set of recipes that included communication among devices both within and without the household using the form shown in Figure 6-6. Though these were not reflective of the functionality demonstrated in OSCAR1 or its successor, OSCAR2, it did reflect functionality that OSCAR was designed to ultimately support, and whose desirability was being assessed for possible future development. In addition, I asked to what extent the addition of the ability to share devices and media across homes added value to OSCAR. Finally, I asked if they believed that the ability to share *recipes* would add value to OSCAR and what role, if any they believed that they would play in recipe sharing (i.e., would they be likely to produce or consume shared recipes?).

6.2.5.3 Form factor and ownership of OSCAR

As mentioned in Chapter 5, the tablet form factor was selected because of its similar properties to existing remote controls but also because of its generous display size. Each participant was asked to react to OSCAR's form factor, and also to comment on what role they could see a device like OSCAR playing in their household.

6.2.5.4 Computer and system adminstration experience

Finally, a set of questions were asked to help characterize each participant's computer expertise and the role they play in home system administration both for themselves and for friends and family. These questions were intended to augment the demographic information we had already collected from each participant during the initial screening phase, which included age, education level, media consumption habits, and type of employment. The information received about each participant from this portion of the interview is reported along with the demographic data in Appendix F.

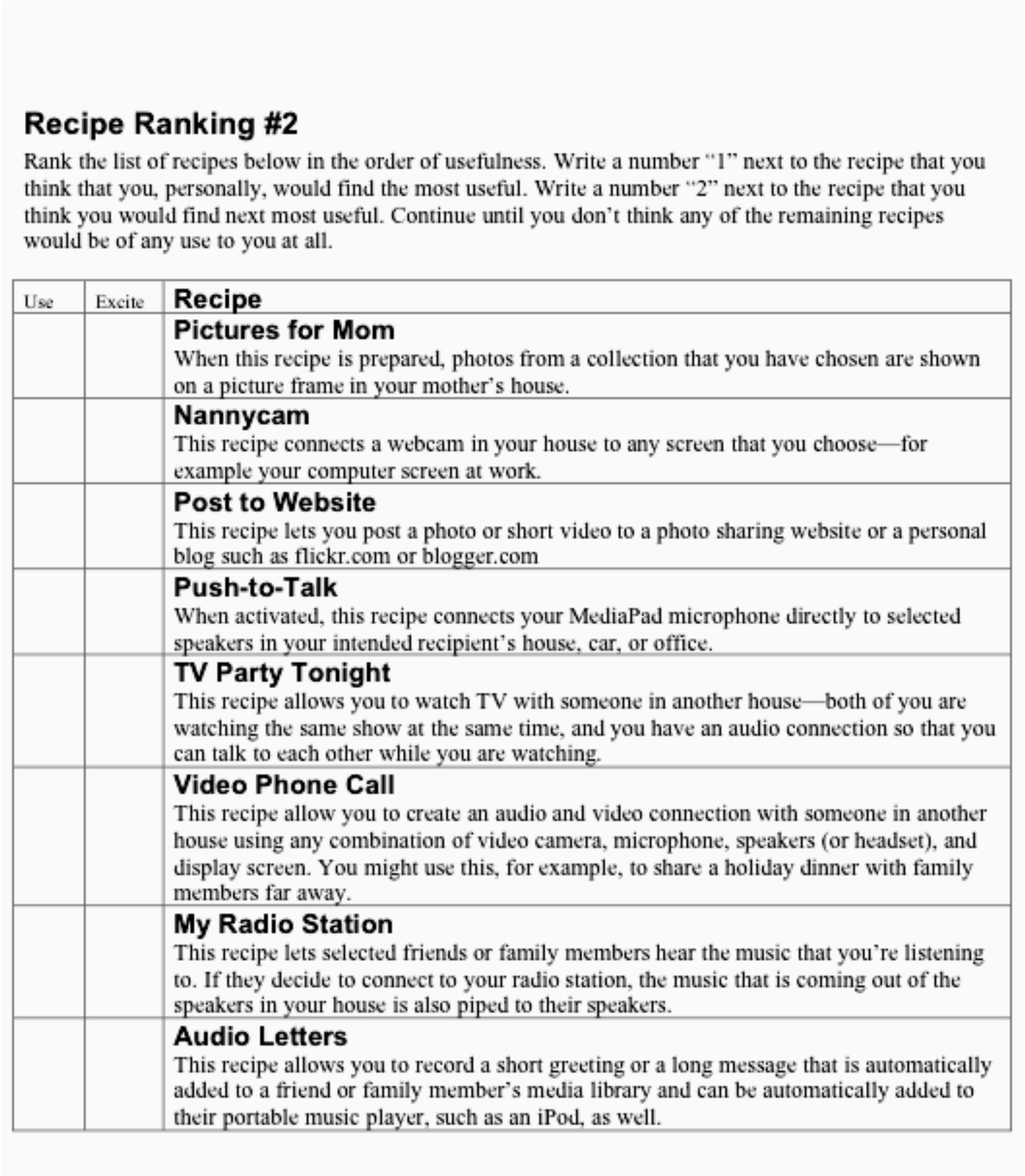


Recipe Ranking

Rank the list of recipes below in the order of usefulness. Write a number "1" next to the recipe that you think that you, personally, would find the most useful. Write a number "2" next to the recipe that you think you would find next most useful. Continue until you don't think any of the remaining recipes would be of any use to you at all.

Use	Excite	Recipe
		Show Off Home Movies When prepared, this recipe allows you to use the MediaPad to choose from a collection of home movies stored in your Media Library. Each time you select a movie, it is shown on the Living Room TV.
		Double Couch Potato When prepared, this recipe allows you to watch two TV shows at once—one on the Living Room TV and the other on any other screen that is available in the Living Room (e.g. a picture frame or the screen of a laptop computer).
		Favorite Photos When prepared, this recipe shows photographs on the Front Door Picture Frame. If you provide it with a list or a collection of photos, it displays a new photo from that list every 30 minutes.
		Room Monitor (a.k.a. Baby Monitor) This recipe connects a microphone in one room to the speakers in another room so that you can hear what is going on in the first room. For example, you could use this to monitor whether a child is sleeping or awake while you are working in another room.
		Who's at the Door? This recipe displays the front door camera on any display in your house. You are prompted to choose the display when you prepare the recipe.
		Leave a Message at the Beep When prepared, this recipe allows you to record a voice message for another member of your household. The message is stored on the media server where the recipient can access it later, and they can be notified by phone, text message, or email that a message is waiting for them.
		Intercom This recipe allows you to speak to someone in another room by connecting the Media Pad's microphone to the speakers in that room. When the recipe is prepared, you are prompted to choose the room you want to talk to.
		Music to Clean By When this recipe is prepared, music is automatically chosen from a playlist or collection and played on the Kitchen Speakers. As you move from room to room, you can use the Media Pad to change the speakers that the audio is piped to.
		Weather and Traffic When prepared, this recipe plays the local weather and traffic radio station on the kitchen speakers and shows video of live traffic cameras and Doppler radar feeds on the Kitchen Screen.

Figure 6-5: The list of recipes presented to users for ranking and discussion. Participants ranked the recipes by assigning each recipe a number from 1-9 in the "Use" column, with 1 being the most useful recipe and 9 being the least. In addition, participants could place an "X" in the "Excite" column next to any recipe that they would be excited to try but that they were not sure would necessarily be immediately useful. The purpose of this study instrument was to present all participants with the same set of recipes to determine whether different users would be likely to use OSCAR in different ways.



Recipe Ranking #2

Rank the list of recipes below in the order of usefulness. Write a number "1" next to the recipe that you think that you, personally, would find the most useful. Write a number "2" next to the recipe that you think you would find next most useful. Continue until you don't think any of the remaining recipes would be of any use to you at all.

Use	Excite	**Recipe**
		Pictures for Mom When this recipe is prepared, photos from a collection that you have chosen are shown on a picture frame in your mother's house.
		Nannycam This recipe connects a webcam in your house to any screen that you choose—for example your computer screen at work.
		Post to Website This recipe lets you post a photo or short video to a photo sharing website or a personal blog such as flickr.com or blogger.com
		Push-to-Talk When activated, this recipe connects your MediaPad microphone directly to selected speakers in your intended recipient's house, car, or office.
		TV Party Tonight This recipe allows you to watch TV with someone in another house—both of you are watching the same show at the same time, and you have an audio connection so that you can talk to each other while you are watching.
		Video Phone Call This recipe allow you to create an audio and video connection with someone in another house using any combination of video camera, microphone, speakers (or headset), and display screen. You might use this, for example, to share a holiday dinner with family members far away.
		My Radio Station This recipe lets selected friends or family members hear the music that you're listening to. If they decide to connect to your radio station, the music that is coming out of the speakers in your house is also piped to their speakers.
		Audio Letters This recipe allows you to record a short greeting or a long message that is automatically added to a friend or family member's media library and can be automatically added to their portable music player, such as an iPod, as well.

Figure 6-6: Another list of recipes that were presented to study participants for ranking. In contrast to the list shown in Figure 6-5, the recipes in this second set involved communications with devices or services outside the home network. Again, participants ranked the recipes from 1-8 in the "Use" column according to their perceived usefulness to them, and could place an "X" in the "Excite" column next to any recipe that they would be excited to try. While this functionality was not present in the versions of OSCAR used in the user tests, its addition had been designed for future iterations.

6.3 Results of the OSCAR1 Study

Users' task performance revealed significant deficiencies in the usability of OSCAR1. None of the users were able to accomplish all of the tasks without at least some help. The results are shown in Table 6-1 and Table 6-2. A serious intervention or "bailout" was required for 4 of the 9 users at least one time during their session, and 3 of those required multiple bailouts. Moreover, many of the solutions arrived at by the participants were, in the research team's assessment, at least somewhat incorrect.

		participant									*	!	!!
		1	2	3	4	5	6	7	8	9			
task	1		!!	*						!!	1	0	2
	2		*								1	0	0
	3a	*					*			*	3	0	0
	3b	!	*			*				*	3	1	0
	3c				*	*			!!		2	0	1
	4a	!	*						!!		1	1	1
	4b	!	!!				*		!!	*	2	1	2
help required		7	9	1	1	2	2	0	9	6	13	3	6

* = reveal system state or call attention to UI element

! = told to take action, !! = told to take sequence of actions

Table 6-1: The amount of help required by each user to complete each task in OSCAR1. The individual "help required" score for each user is computed as the sum of the amount of help required for each task, where * = 1, ! = 2, and !!= 3.

		participant									◢	✖
		1	2	3	4	5	6	7	8	9		
task	1		✖									1
	2											
	3a	◢	◢			◢	◢	◢			5	
	3b		◢						◢		2	
	3c		◢				◢		✖		2	1
	4a	✖	◢					◢	✖	◢	3	2
	4b	✖	✖	◢		◢	◢		✖	◢	4	3
degree wrong		5	8	1	0	2	3	2	7	2	16	8

◢ = partially incorrect, ✖ = completely incorrect

Table 6-2: The correctness of each OSCAR1 user's solutions to each task. Individual users' "degree wrong" is computed by assigning 1 point to partially incorrect solutions and 2 points to completely incorrect ones.

These results indicated that we needed to make improvements to the OSCAR user interface. After analyzing the task sessions, we were able to identify 174 incidents that indicated at least some degree of user confusion or dissatisfaction. We boiled these down to 52 usability issues of which we deemed 14 to be high priority. Most of these were fixed in the development of OSCAR2. Examples of a few of these usability problems are shown in Figure 6-7 and Figure 6-8.

Figure 6-7(a) shows one of the most significant usability problems, which was the label “Preparing (Ready Soon)” in the Connection Detail screen when a new connection was in the process of being configured from a recipe. Several users interpreted this label as indicating the system was busy with something and that, therefore, they should wait until it was finished before proceeding. This was an unfortunate choice of language that was based on the desire to reinforce the “recipe” metaphor but conflicted sharply with users’ interpretation.

Figure 6-7(b) shows a frustration that was experienced by several users: it was not possible to select a component for inclusion in a connection when the recipe from which it was being prepared did not have criteria that matched the desired component. Most commonly this was experienced because of an error in setting up the recipe that was realized during the “prepare” phase. OSCAR1 did not provide adequate means of recovering from such errors.

Several users did not recognize that they were being asked to press “choose” after making a selection using the component browser, as shown in Figure 6-7(c).

In this case, a more clear design along with improved instructions was required to inform people that and explicit “choose” was required after making a selection. This step could not be eliminated without significant redesign because the browser was designed to allow users to select aggregate components as well as individual components. When an aggregate component is selected, it also expands to the right to reveal its contents, which can also be chosen. This ambiguity allows for more fluid browsing but requires that the user explicitly press “choose” to select an item.

Finally, one usability issue arose from the combination of the form factor with the fact that the user study participants were seated on a couch during the sessions. While such an arrangement simulated a realistic usage scenario for a device intended for configuration and control of home media devices, it had the unintended side effect of preventing certain users from being able to see the lower portion of the screen. In particular, users’ stomachs that protruded above their waistbands obscured up to a quarter of the screen, and it did not occur to these users that they might be missing important information that was being occluded. As a result, certain tasks were very difficult for such users to complete. As shown in Figure 6-8, buttons such as the “New Recipe” button were frequently hidden, and certain tasks, such as Task 3, were nearly impossible to complete without using this button. We came to refer to this problem as the “belly issue” and strove to fix the problem in OSCAR2.

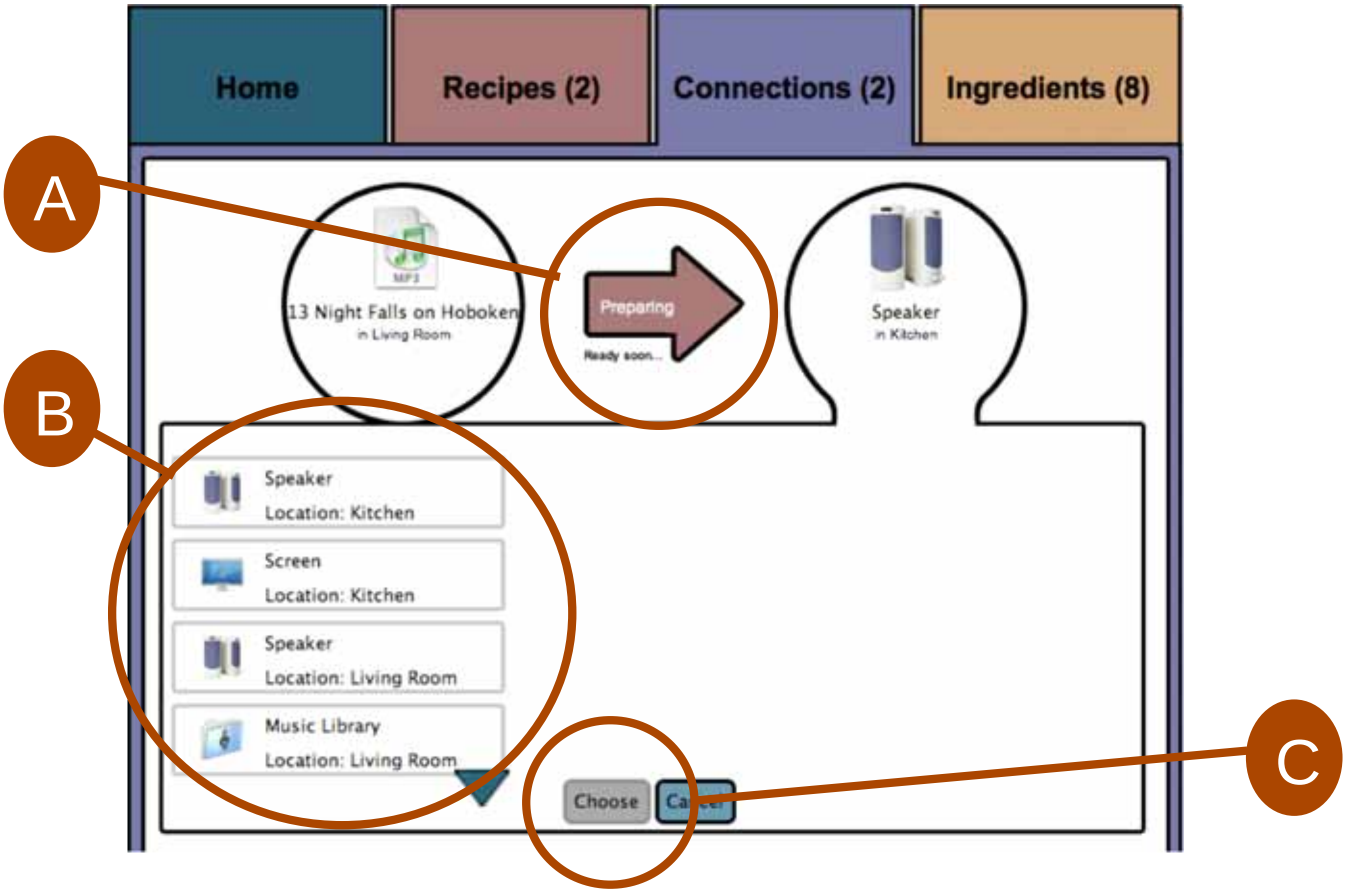


Figure 6-7: Three of the most significant usability problems with OSCAR1 were located on the Connection Details screen. The language used in (A) when a new connection is being prepared from an existing recipe was confusing to many users. The label "Preparing (Ready soon…)" was interpreted to mean that the system was busy, rather than that the user was expected to browse and select a component in the browse dialog below. The list in (B) would only show components that had been predetermined in the originating recipe, whether by explicit inclusion or by defining abstract criteria. There was no way for users to override this list and select some other component, which led to difficulties. Additionally, several users failed to notice that the "Choose" button became enabled once a component had been selected, and that they were expected to tap "Choose" in order to complete the configuration of the connection.

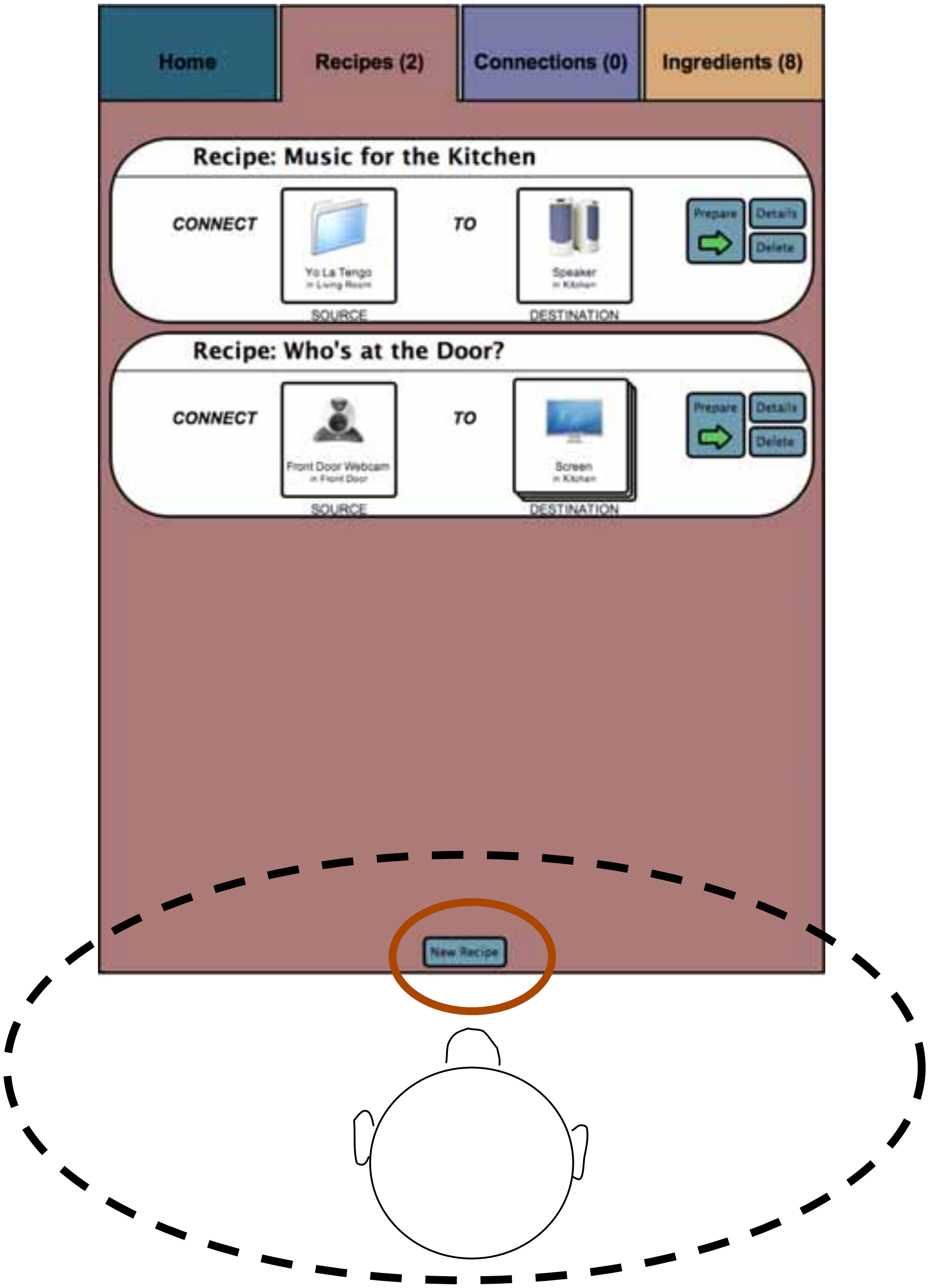


Figure 6-8: The "Belly Issue" experienced by some of the OSCAR1 participants. Because the OSCAR tablet was used on a couch, some users held the tablet in such a way that they were unable to view the entire screen. In particular, the lower portion of the screen was occluded by some users' stomachs and clothing that protruded over the edge of the device when it was resting against their waists.

6.3.1 Screen-by-screen assessment results

Participants' assessment of the individual screens was also helpful for prioritizing the usability issues at a higher level. The screen-by-screen exercise gave fairly clear pointers to which screens caused the most trouble and some of the reasons for that. It also indicated which screens were more clear and useful for accomplishing the tasks. For example, the OSCAR1 Recipe Details screen received negative feedback from 6 of the 8 participants who participated in this exercise (one skipped this part due to time), and positive feedback from only one. Much of the negative feedback focused on the "selection rules" portion of the screen, as shown in Figure 6-9, and several of the comments indicated that these options were confusing or unclear. Other comments complained that it wasn't clear where it was possible to tap to make selections in this screen, a problem that applied to other screens as well but appeared to be particularly acute here. Additionally, users highlighted difficulties with the process of selecting components after "preparing" a recipe (Figure 6-10, left), and criticism of the Ingredient Detail (Figure 6-10, right) screen, which was seen as being so devoid of information in typical usage that its purpose was doubtful. On the other hand, several users had praise for the three list screens (Recipes, Connections, and Ingredients, see Figure 6-11) because they felt that these screens gave them perspective about what was going on in the system and helped them understand what the system could do.

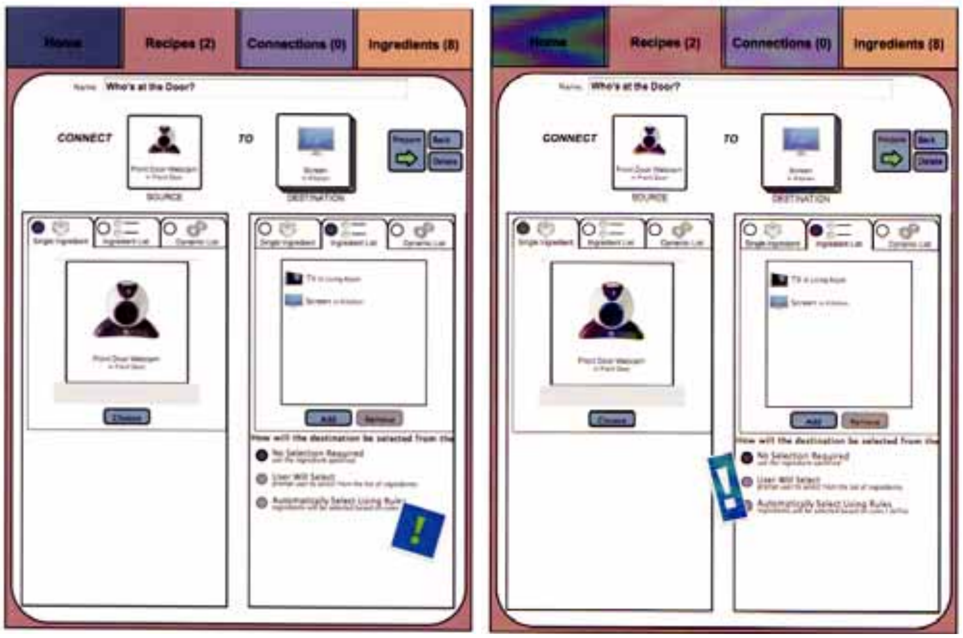


Figure 6-9: The "Recipe Details" screen, and in particular the "Selection Rules" section of the screen, was flagged by several users as a difficult part of the OSCAR1 user interface.

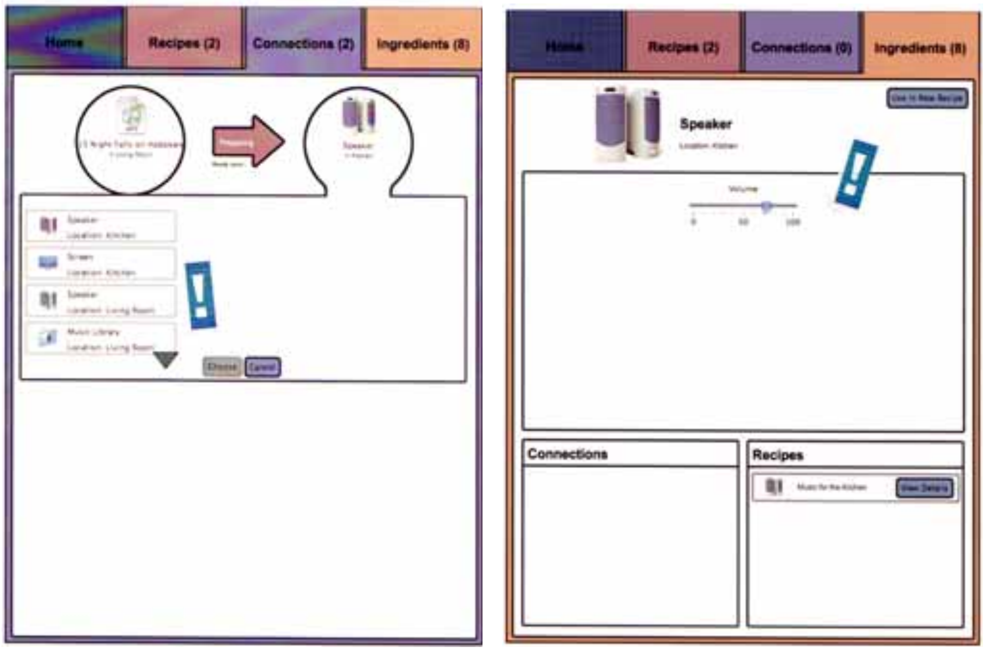


Figure 6-10: Both the "Connection Details" and "Ingredient Details" screens were also flagged by participants as troublesome parts of the OSCAR1 user interface.

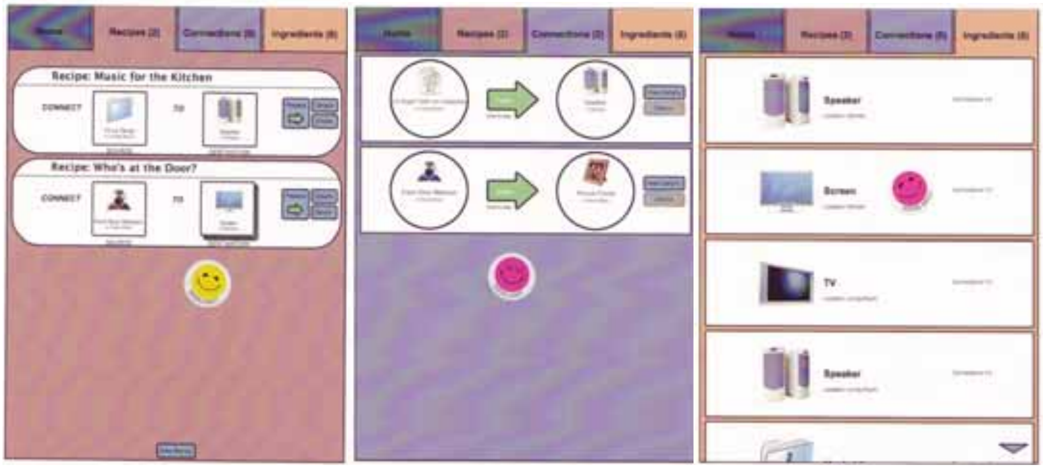


Figure 6-11: Each of the "List" screens ("Recipes," "Connections," and "Ingredients") were identified by multiple users as particularly helpful portions of the OSCAR1 user interface.

6.3.2 Reactions to the Recipe Metaphor

One reason for poor performance may have been confusion about the language used in the user interface. All 9 participants expressed at least some degree of dislike for the term "recipe," and more than half expressed *strong* disdain. One participant expressed a common sentiment: "[I would] prefer to use technical terms. Let's just call it what it is." This statement and others conveyed a sense that we were trying to mislead users into believing that our application was easier to use than it really was, and this attempted deception was met with hostility. Furthermore, one particular aspect of our selected terminology created extreme confusion: our choice of the term "prepare" to denote the action of invoking a recipe. This term did not match user's expectations about how to achieve the effects they desired, and in several instances they would say aloud that they were looking for a *Play* or *Start* button while they overlooked the *Prepare* button repeatedly.

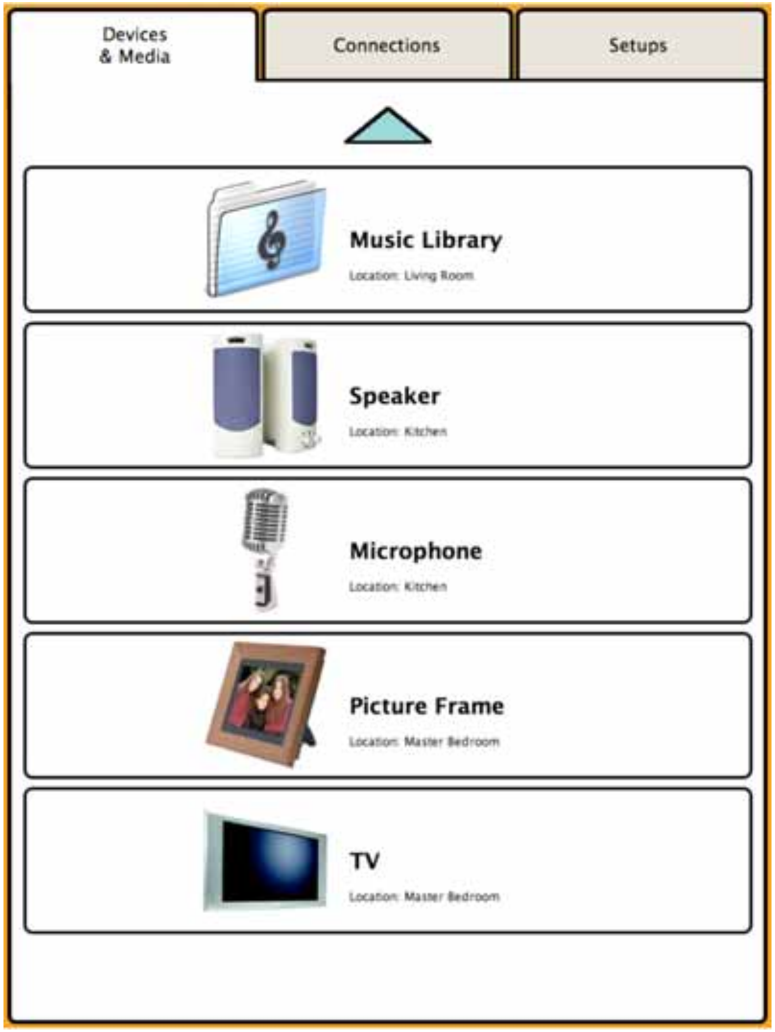
6.3.3 Subjective Impressions and Interview Results

The results of the other aspects of the OSCAR1 user study, including the subjective assessments of usability, recipe and activity preferences, and general interview results are presented after the description of OSCAR2 and the second user study. Where the results are expected to differ across the two phases, for example in the subjective assessments of usability and preference, the data is presented as a comparison to highlight the differences between the two versions. Where the results are not dependent on the version of the prototype used, for example the interview results regarding activity/recipe preference and attitudes towards sharing, the results from both phases will be presented together.

6.4 The OSCAR2 Prototype

The feedback from the OSCAR1 study was used to make changes to the interface for OSCAR2. As described earlier, the only pre-determined difference between the prototypes was a decision to explore the activity-oriented worldview in OSCAR1 and the device/media-oriented worldview in OSCAR 2. This was expressed in the interface by changing the entry point into the system. In OSCAR1 the first screen showed a list of recipes, and in OSCAR2 that was changed so the first screen showed a list of devices and media services (Figure 6-12). We expected that the first screen that was seen would be immediately regarded as the "home" screen and would be the starting point for each set of interactions with OSCAR, a distinction reinforced by reordering the navigation tabs so that the point of entry (*Recipes* or *Devices & Media*) was always the left-most tab.

The second significant change was in language. Based on the negative reactions to the term "recipe" and the related usability problem, the term "recipe" was changed in favor of "setup" to describe pre-defined connections, and the operation of invoking a recipe/setup changed from "prepare" to "run" (Figure 6-13). Based on the numerous problems experienced with the Connection Details screen (see again Figure 6-7 and Figure 6-10), a number of improvements were made to make the flow of instantiating a new connection from a predefined Setup more clear, and interacting with it afterwards more flexible (see Figure 6-14). A number of lesser user interface improvements were also made, including changes to the graphic design to clarify which regions of the screen were clickable and which weren't, and changes to the browser dialog to make it appear more clearly modal. A number of bugs were also fixed that affected the system's performance and reliability.



Music Library
Location: Living Room

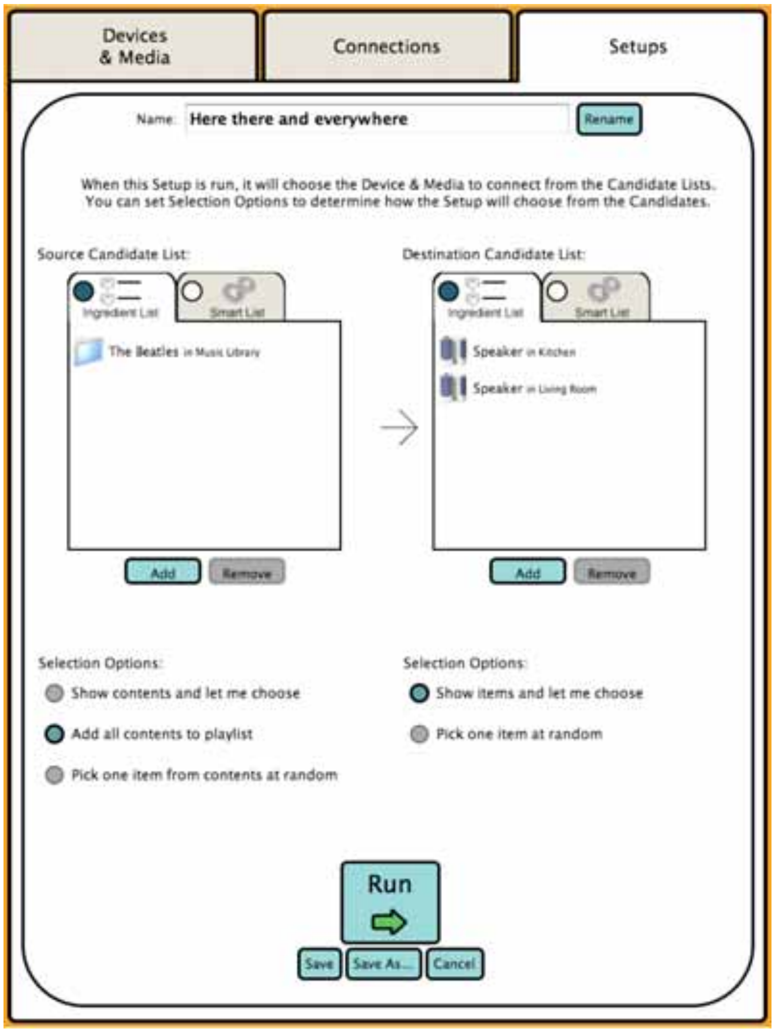
Speaker
Location: Kitchen

Microphone
Location: Kitchen

Picture Frame
Location: Master Bedroom

TV
Location: Master Bedroom

Figure 6-12: The Devices & Media screen offers a list of Obje services and devices that have been discovered by OSCAR.



Name: Here there and everywhere

When this Setup is run, it will choose the Device & Media to connect from the Candidate Lists.
You can set Selection Options to determine how the Setup will choose from the Candidates.

Source Candidate List:

- The Beatles in Music Library

Destination Candidate List:

- Speaker in Kitchen
- Speaker in Living Room

Selection Options:

○ Show contents and let me choose
● Add all contents to playlist
○ Pick one item from contents at random

Selection Options:

● Show items and let me choose
○ Pick one item at random

Figure 6-13: This view of the Setup Detail screen shows an edited setup that will allow the user to play all songs by the Beatles on either of two speaker sets, which will be selected by the user when the setup is run.

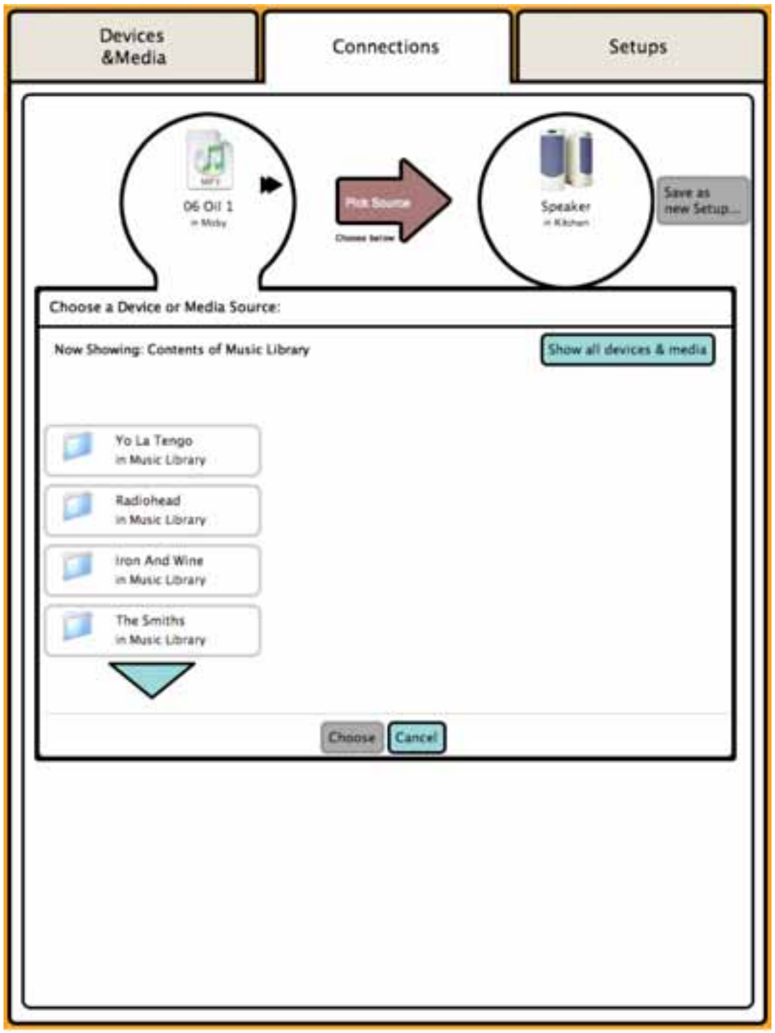


Figure 6-14: This view of the Connection Detail screen shows a connection in the process of being set up. This screen underwent significant improvements between OSCAR1 and OSCAR2.

6.5 Results of the OSCAR2 Study

The usability of OSCAR2 was much improved over OSCAR1. The amount of help required, shown in Table 6-3, decreased from 22 incidents (including 9 bailouts of which 6 were severe) to 13 (including 1 bailout that was not severe). All users but one were able to complete all tasks without any serious help. We attribute this improvement to the changes in to the language (i.e., the replacement of "recipe" with the "setup" terminology) and to the superior usability of the device-oriented interface model, though we are unable to say how much each improvement contributed to the overall improvement in users' performance.

On the other hand, the number of errors (shown in Table 6-5) with OSCAR2 was only one fewer than those made by OSCAR1 users (23 vs. 24). A higher fraction of these were minor errors (20/23 vs. 16/24), most of which were users accepting the default options when they should have changed them. For example, in Task 3b where users made a setup that allowed the webcam image to display on either the picture frame or TV, users needed to change the default option from *pick one item at random* to *show items and let me choose*. Another common error was to define only half of the setup—specifying either source or destination—before running it, and then completing the connection when prompted in the connection initialization process. Even when these errors were made, users seemed to be unaware that they had made a mistake. Often they were able to make the desired connection and thus experience the desired effect (e.g., viewing the webcam image on the TV) with little or no additional effort, so it seemed to them that they had correctly completed the task.

		participant											
		10	11	12	13	14	15	16	17	18	*	!	!!
task	1										0	0	0
	2										0	0	0
	3a			!							0	1	0
	3b					*		*			2	0	0
	3c					*					1	0	0
	4a	*							*		2	0	0
	4b	*	*	*	*	*	*		*		7	0	0
help required		2	1	3	1	3	1	1	2	0	12	1	0

* = reveal system state or call attention to UI element
! = told to take action, !! = told to take sequence of actions

Table 6-3: The amount of help required by each user to complete each task in OSCAR2. The individual "help required" score for each user is computed as the sum of the amount of help required for each task, where * = 1, ! = 2, and !!= 3. The amount of help required for users of OSCAR1 is shown again in Table 6-4 for comparison.

		*	!	!!
task	1	1	0	2
	2	1	0	0
	3a	3	0	0
	3b	3	1	0
	3c	2	0	1
	4a	1	1	1
	4b	2	1	2
		13	3	6

Table 6-4: A summary of the amount of help required to complete each task in OSCAR1. This data is copied from Table 6-1 and presented here for comparison with the OSCAR2 help data presented in Table 6-3. The comparison shows that quite a bit less help was required by users of OSCAR2.

		participant									◢	✖
		10	11	12	13	14	15	16	17	18		
task	1								◢		1	
	2											
	3a											
	3b				◢				◢		2	
	3c			◢	◢	◢	◢	◢	◢	◢	7	
	4a		◢	✖	✖	◢	◢	◢	✖	◢	5	3
	4b					◢	◢	◢	◢	◢	5	
degree wrong		0	1	3	4	3	3	3	6	3	20	3

◢ = partially incorrect, ✖ = completely incorrect

Table 6-5: The correctness of each OSCAR2 user's solutions to each task. Individual users' "degree wrong" is computed by assigning 1 point to partially incorrect solutions and 2 points to completely incorrect ones.

		◢	✖
task	1		1
	2		
	3a	5	
	3b	2	
	3c	2	1
	4a	3	2
	4b	4	3
		16	8

◢ = partially incorrect, ✖ = completely incorrect

Table 6-6: A summary of the correctness of users' solutions to each task in OSCAR1. This data is copied from Table 6-2 and presented here for comparison with the OSCAR2 correctness data presented in Table 6-5. The comparison shows that users' solutions did *not* improve much between OSCAR1 and OSCAR2, though the severity of the errors did decrease somewhat.

6.5.1 Analysis of breakdowns

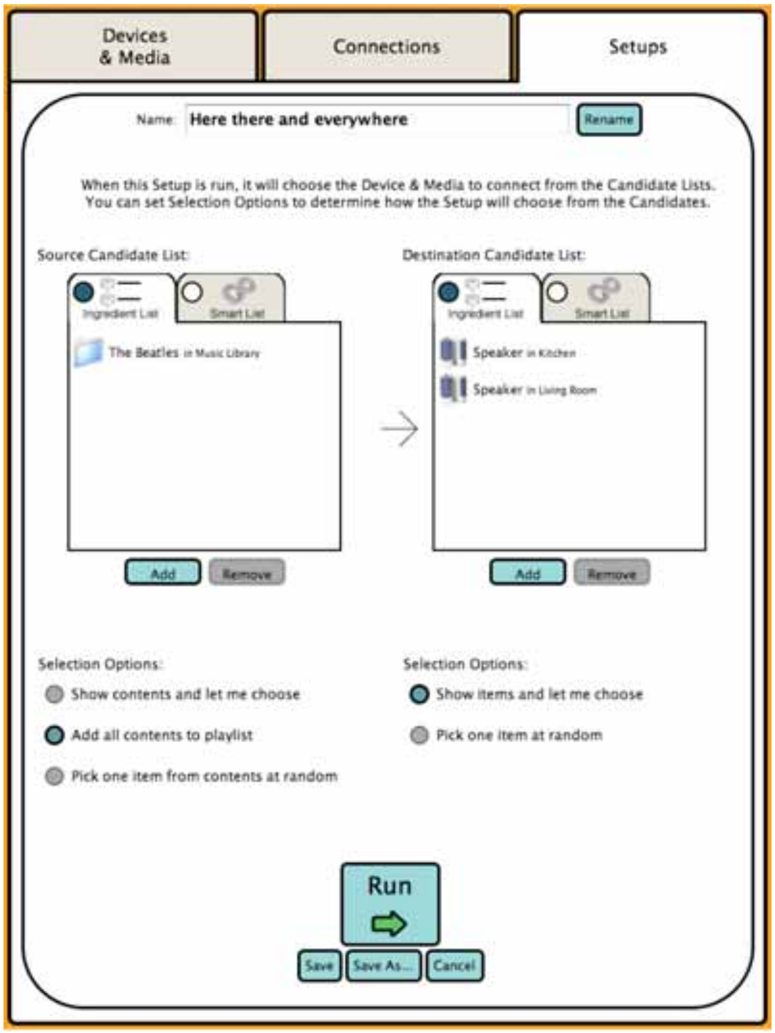
The most remarkable shortcomings of OSCAR2 were the amount of help required for Task 4b and the number of incorrect solutions for tasks 3b, 4a, and 4b.

6.5.1.1 Help required for Task 4b

Analyzing the help required for Task 4b reveals that several of the difficulties were quite minor and may have been overcome by the user if help had not been readily available in the form of the study hosts. Our protocol was such that we encouraged participants to complete the tasks as if we weren't there, but that we would help them get unstuck if requested. Recall that Task 4b asks the user to modify the Setup they had just created in Task 4a. Task 4a requested the user to "Make a setup that will allow [their roommate] to play all of the songs by the Beatles in the kitchen," and 4b asks them to refine this so that "[their roommate] can pick the artist and speakers when they activate the setup." Despite our intention that participants would modify the Setup they had just created, many of them started over from scratch and made a new Setup to satisfy Task 4b.

Of the seven participants who required help on Task 4b, four (participants 10, 11, 12, and 13) ended up with correct solutions. In each of these cases, the user either had the right solution or was on the path to the right solution, but became apprehensive at some point and asked for help.

Two users who ultimately came up with correct solutions (participants 10 and 12) got stuck on the Setup Details screen when they were trying to decide between the options "Show items and let me choose," "Add all items to playlist," and "Pick one item at random" (see Figure 6-15). In both cases, the user had guessed correctly that "Show items and let me choose" would produce the correct functionality, however they were sufficiently apprehensive about their choice that they would not proceed without help. In each case, the user was simply encouraged to try it out to see if works. Again, without a ready source of help, these users may have eventually overcome their apprehension and run the Setup to see what happened. In that case, they would have realized that their selection had been correct.



Name: Here there and everywhere

When this Setup is run, it will choose the Device & Media to connect from the Candidate Lists.
You can set Selection Options to determine how the Setup will choose from the Candidates.

Source Candidate List:

- The Beatles in Music Library

Destination Candidate List:

- Speaker in Kitchen
- Speaker in Living Room

Selection Options:

○ Show contents and let me choose
● Add all contents to playlist
○ Pick one item from contents at random

Selection Options:

● Show items and let me choose
○ Pick one item at random

Figure 6-15: The "Setup Details" screen caused a number of difficulties for users. One user did not understand the function of the "Run" button and was unable to complete Task 4b without help despite having created a correct Setup. Two users required help to choose among the "Selection Options" for one or both of the Setup slots. Additionally, numerous minor errors were attributable to this screen. Many users failed to choose the correct Selection Option(s) or even notice that they were available. Other users failed to select items for either or both of the "Source Candidate" and "Destination Candidate" lists and ended up running a partially defined Setup.

Two other users with ultimately correct solutions (participants 11 and 13) required help after creating a correct Setup and hitting "Run." During the subsequent step of choosing the components to use in the Setup (in the case of Task 4b, this meant choosing the artist from the Music Library and then choosing a specific set of speakers), they got stuck because they didn't realize they were being presented with a modal dialog box that forced them to select a component and then hit Choose (see Figure 6-16). Instead they believed OSCAR had stopped responding to them and that we needed to restart the prototype or otherwise intervene. In each case, the user study host (Ame Elliott) gave them a hint that got them back on track ("You need to complete the selection first," and "What would happen if you hit 'Choose?'"). In each case, it is not clear whether the user would have been able to solve the problem themselves.

The remaining three users who required help (participants 14, 15, and 17) did not generate correct solutions even with help.

One such user (participant 14) struggled with selecting multiple artists from the Music Library to include in the Setup (which could be accomplished by selecting the entire Music Library or by selecting multiple individual artists). She knew that this was her goal but was unable to figure out how to do it because of the way that Aggregate components are treated in the OSCAR browser: the same operation is used to expand Aggregates as is used to select them. When the cursor is on an Aggregate, selecting "Choose" will select all of the contents. If the cursor is on one of the Aggregate contents, selecting "Choose" will select only the highlighted item (again, see Figure 6-16). Since tapping an Aggregate such as the Music Library would always expand the contents, this user was unable to prevent herself from selecting one of the expanded items. Eventually, despite encouragement and hints, she created a setup that played one song.

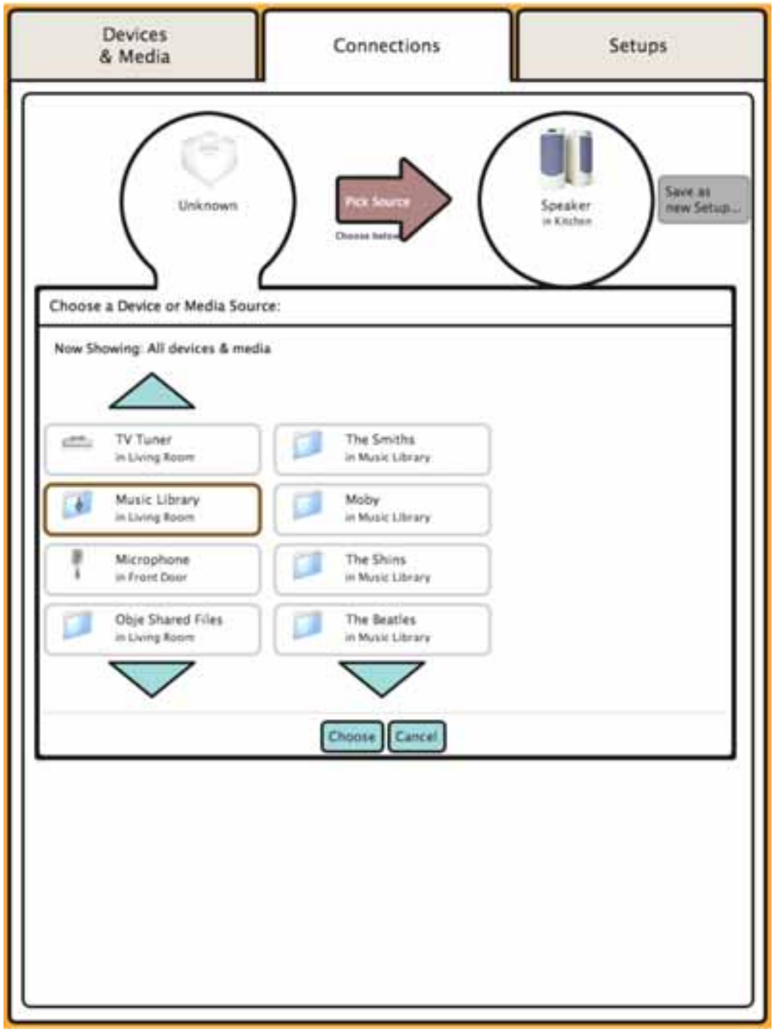
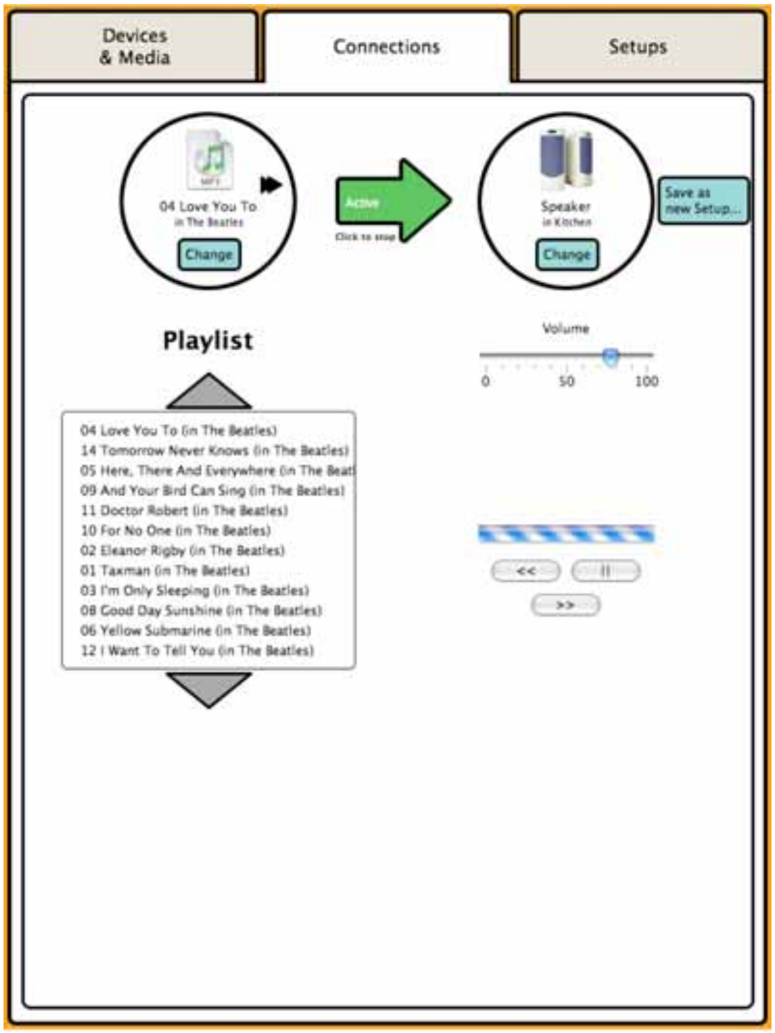


Figure 6-16: Users struggled with the Connection Details screen, especially when "running" a Setup with a slot whose Selection Options were set to "Show items to me and let me choose." Two users required help because they did not realize that they were required to tap "Choose" after making a selection in the Browser overlay. This figure also highlights another difficulty that users encountered. A number of users did not realize that, with the screen in the state shown here, they could tap "Choose" to select all of the contents of the "Music Library. "Rather, they believed that because the contents had been expanded to the right when the Music Library had been initially chosen, they were obligated to select among one of the expanded items. These users struggled with tasks like 4a and 4b that required them to select a collection of music to be the source of a Setup or Connection.

Another user (participant 15) managed to create the correct Setup but did not know what to do next. She was blind to the “Run” button on the Setup Details screen (refer again to Figure 6-15), and returned to the Device & Media screen several times to start over from scratch. She created several versions of the Setup and eventually ran a Setup that had only one song as the source even though she had correctly populated the source slot with the Music Library in a previous version.

Yet another user (participant 17) believed that he had accomplished the task when he created a connection between a song by the artist Beck and the living room speakers, which is what the Task instructions had suggested as a means of testing the correctness of the Setup. However, this user had not actually created a Setup, but only a one-off connection. When it was pointed out to him that his connection would not be able to be used by his roommate, as was stipulated in the task, he seemed confused by the instructions and proceeded to make a Setup with the Music Library plus one (stray) song in the source slot and nothing in the destination slot. He then ran the Setup and made the rest of the selections (choosing the artist and set of speakers) in the Connection Details screen. As shown in Figure 6-17, it is possible to choose other devices and media from within the Connection Details screen, so even if the initial connection is partially specified or even wrong.



04 Love You To (in The Beatles)
14 Tomorrow Never Knows (in The Beatles)
05 Here, There And Everywhere (in The Beat
09 And Your Bird Can Sing (in The Beatles)
11 Doctor Robert (in The Beatles)
10 For No One (in The Beatles)
02 Eleanor Rigby (in The Beatles)
01 Taxman (in The Beatles)
03 I'm Only Sleeping (in The Beatles)
08 Good Day Sunshine (in The Beatles)
06 Yellow Submarine (in The Beatles)
12 I Want To Tell You (in The Beatles)

Figure 6-17:The Connection Details screen allows the user to select a different source or destination for the displayed connection at any time by tapping on the "Change" button below the currently selected item. This means that users can make effective use of even partially or incorrectly defined Setups by adjusting their parameters after they have been "Run."

6.5.1.2 Incorrect Solutions to Tasks 3c, 4a, and 4b

In analyzing the various incorrect solutions to Tasks 3c, 4a, and 4b, there are only a handful of reasons why the solutions were deemed incorrect. Of the catastrophic failures, all occurred on Task 4a. One of them (by participant 17) was discussed above, and another user (participant 13) shared the same problem: these users never created a Setup, but rather made a connection between a song and the speakers and was satisfied with the results. These failures occurred at least in part because users believed they had achieved success when they were able to hear the music they had selected playing from the speakers they had selected. It is not entirely clear whether they misunderstood the task instructions, the Setup construct, or both. In any case, they did not distinguish between making a one-off connection to be executed immediately and creating a reusable Setup that they or a household member could access later (refer again to Figure 6-17).

The remaining catastrophic failure occurred because the user (participant 12) created a Setup consisting of one song as the source and nothing as the destination. After running the Setup, she was able to select speakers and play the song, but never figured out how to play a second song as required by the task instructions. This failure was connected to the aforementioned confusion about selecting an Aggregate versus selecting an item contained within the Aggregate (refer again to Figure 6-16).

In addition to the three catastrophic failures on Task 4a, there were 17 minor failures on 3c, 4a, and 4b. These are significant because these are all of the tasks that require the user to create and then edit a Setup, and indeed all of the errors were related to properly constructing Setups and were tied directly to the Setup Details screen (refer again to Figure 6-15). Eleven of the errors resulted from leaving the default selection criteria for the destination slot as "Pick an item at random” rather than changing the criteria to “Show items to me and let me choose.” While the latter would perhaps have been a reasonable default choice for all new Setups, it had been left as “random” for the express purpose of seeing whether users could understand these selection criteria and override the defaults. The number of errors observed indicates that, indeed, these options were difficult to understand. The remaining six errors all were related to selecting an incorrect value for one or both of the Setup candidate lists. One user created an empty setup and ran it, two users left the destination slot empty, and three users selected a single song for the source rather than a collection of songs. In all of these cases the user was able to adapt the resulting connection after running the Setup to do what he or she expected, and in most cases did not even realize that a mistake had been made. I believe that if users were creating and using Setups for regular use in their homes, they would overcome many of these minor errors after a few days. Given the short time span of the user test, however, there was not much opportunity to experience the disadvantages of creating an incorrect Setup, and so the motivation to learn the correct techniques was not very high.

These various difficulties, unfortunately, do not point to a single flaw with OSCAR, but rather a host of relatively small interaction and usability errors that would need to be fixed before OSCAR could be deployed to a wide variety of users. The fact that these errors emerged in OSCAR2 may be indicative of the severe usability issues that existed with OSCAR1 that masked these lesser issues.

On the other hand, these problems do point to the conceptual difficulty that users have contending with the notion of Setups and in particular navigating the tradeoff between work done in advance and work done at the time of use. A longer term study of OSCAR would need to be conducted to see if users become motivated to learn how to make more effective Setups after spending some time using OSCAR. It may well be that novice users would be content manually connecting devices and media together for quite some time before they decide to invest effort in mastering Setups. The fact that OSCAR2 makes is fairly easy to create one-off connections may actually reduce users' inclination to create accurate Setups, and this fact presents an interesting tension between immediate and future-looking configuration that is inherent in end-user composition and merits further investigation.

6.5.2 Additional usability issues

Another class of usability issues that did not have a measurable effect on performance but was apparent in the context of the individual study sessions has to do with the generic nature of the operations and abstractions supported by OSCAR. An apparently minor subtask that gave many users problems was being asked to remove the Front Door Webcam image from the Front Door Picture Frame. The expected solution to this was to click the arrow in the *Connection Details* screen that was labeled (albeit in small letters) *Click to Stop*. However, this was not apparent to users as it was not clear that "stopping" a still image would remove it from the frame. Not surprisingly, users were more successful when we asked them to "kill" the song that was playing—the term "stop" is seen as a close enough synonym to "kill" in this context, whereas "stop" does not mean "remove" to most people.

Some users solved the problem of removing the webcam image by selecting a different image from the Photo Library and putting this on the picture frame instead. Most users were eventually able, often through trial and error, to understand that the picture could be removed from the picture frame by stopping the connection. The fact that OSCAR was controlling live devices was helpful in this regard, as it was easy for users to try different approaches and see the results of their actions in the environment. Thus, despite some initial awkwardness resulting from the generic language used in OSCAR, most users were able to form an appropriate mental model and perform successfully.

Other user difficulties were common to many applications; for example, some users encountered difficulties because of a lack of "undo" functionality and the inability to select multiple items at once when choosing components for a setup or connection. Several users expected a "back button." All users were affected to at least some extent by the occasionally poor performance of the application combined with inadequate feedback about its current state, causing confusion such as believing the device was sluggish when the interface was waiting for input.

The touchscreen itself was often difficult to use and OSCAR did not provide adequate feedback to acknowledge receipt of an input event.

6.5.3 Screen-by-screen Assessment of OSCAR2

Users' responses to the screen-by-screen assessment again reinforced the usability issues we saw. They also showed that the prototype had made limited progress in certain areas. Not surprisingly, given the foregoing discussion of user difficulties, the Setup Details screen (which had replaced the Recipe Details screen from OSCAR1) continued to meet with negative comments (see Figure 6-18), though the comments were more specific and detailed, and more varied across users, indicating that there was less general confusion and frustration but that certain design details needed to be worked out. There were, again largely positive responses to the list screens, most notably to the Devices & Media List screen (see Figure 6-19), which had replaced the Recipe List as the initial, or "home," screen that greets first-time users. It was notable from users' comments about this screen that they were comforted by the "obviousness" of this screen and that it appeared to make them more comfortable using OSCAR overall.

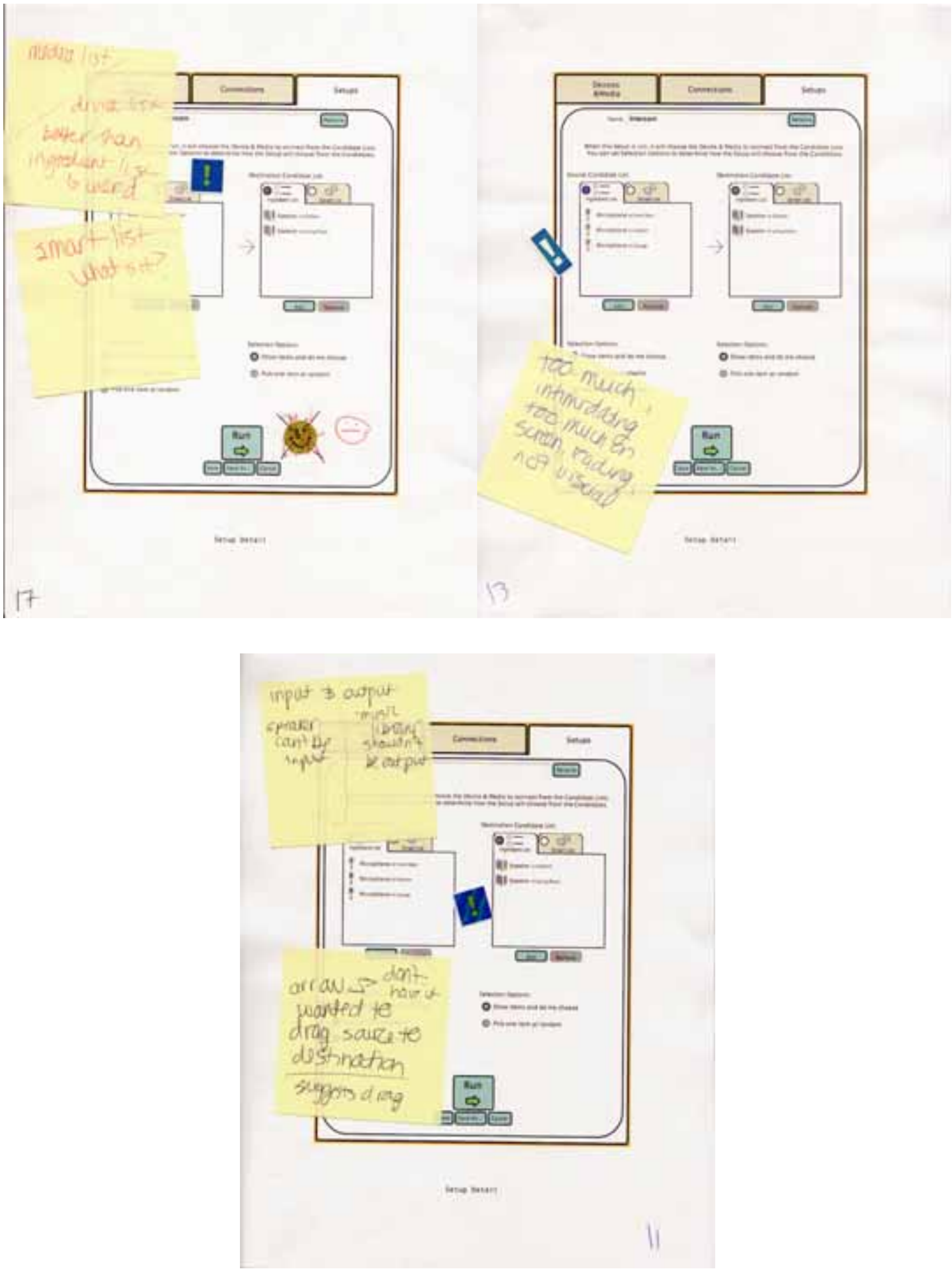


Figure 6-18: Participants' assessments indicated that significant problems continued to exist with the usability of the "Setup Details" (formerly "Recipe Details") screen. However, the comments regarding OSCAR2's Setup Details screen were considerably more detailed and suggested a greater overall understanding of the intent of the screen and the function of each of its controls.

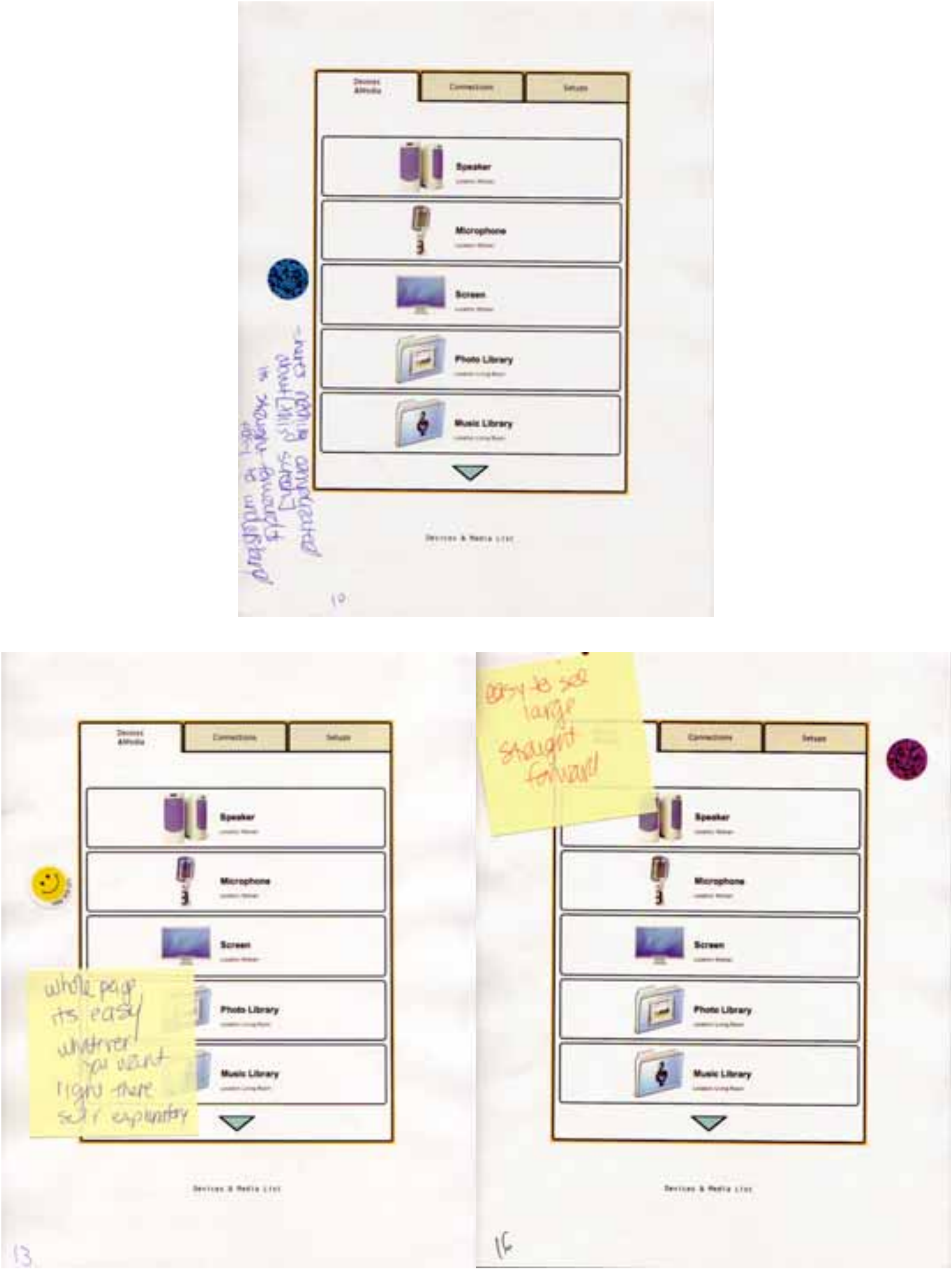


Figure 6-19: As with OSCAR1, each of the List screens received positive comments from users. In particular, the new "Devices & Media" list screen was well received.

6.5.4 Subjective Perceptions of OSCAR1 and OSCAR2

The users' perception of the system's usability improved from OSCAR1 to OSCAR2 as shown in Figure 6-20 and Figure 6-21. The perceptions of OSCAR1 were more widely varied, with some extremely low scores and a couple of extremely high scores. As described above, the System Usability Scale (SUS) [16] was used to assess participants' opinions regarding the usability of OSCAR. The SUS consists of ten statements that yield responses between 1-5 indicating the respondents agreement with each statement. The responses are used to compute an overall usability score between 0-100, with 100 being the highest possible usability rating. The ten SUS questions made up the first ten questions of the post-task questionnaire given to all user study participants (refer again to Figure 6-4). The last seven questions were devised by myself, and were intended to assess various aspects of OSCAR's usefulness and desirability.

On average, users' perception of OSCAR1's usability according to responses to the SUS portion of the questionnaire was lower (55/100 vs. 63/100), though with a much higher variance (26 vs. 12). The difference was not statistically significant based on a one-tailed t-test assuming equal variances. Interestingly, the difference between OSCAR1 and OSCAR2 participants' response to question #3 from the SUS, "I think that I would like to use this system frequently," *was* statistically significant in favor of OSCAR2 (3.3/5 avg, 0.7 stdev) as compared to OSCAR1 (2.3/5 avg, 1.3 stdev) based on a one-tailed t-test assuming equal variances (p

<0.05). This question resembles the subjective preference questions that we asked at the end of the questionnaire (questions 11-17, omitting 14), and the responses to those questions were again significantly in favor of OSCAR 2. OSCAR2 users responded more positively (4.43/5 avg, 0.47 stdev) to these questions than OSCAR1 users (3.59/5 avg, 1.19 stdev). This difference *was* statistically significant based on a one-tailed t-test assuming equal variances (p<0.05). A summary of the responses to the subjective desirability questions is shown in Table 6-7, and the full set of responses to all questionnaire questions, including the SUS and subjective preference questions, are shown in Appendix F.

Question	OSCAR1	OSCAR2	Significant?
11. I would rather use this system than what I use now for interacting with my devices.	3.7	4.3	No
12. I would like to have this system if it were available.	3.7	4.4	Yes (p<0.05)
13. I would recommend this system to a friend.	3.4	4.3	Yes (p<0.05)
15. I think I would enjoy using this system at home.	3.6	4.4	Yes (p<0.03)
16. I think it would be fun to use this system.	3.4	4.4	No
17. I would find this system very useful in my own home.	3.8	4.6	Yes (p<0.05)
Overall Average	**3.6**	**4.4**	**Yes (p<0.05)**

Table 6-7: Average responses to questions regarding subjective preference given by participants in the OSCAR1 and OSCAR2 user studies. In each case, the average reflects responses given on a scale of 1-5, where 5 meant that the user "Strongly Agreed" with the statement. This data shows that OSCAR2 was more acceptable to users than OSCAR1 by a significant margin.

6.5.5 Correlations with Computer and Networking Experience

For both OSCAR1 and OSCAR2, the most experienced users (as indicated by their computer experience and system administration experience) required the least help in order to complete the tasks. In general, the more experienced computer users made fewer errors but there were a few dramatic deviations from this trend in both directions: highly experienced people who made a lot of errors and less experienced people who made few errors. In these cases, the skilled users who made errors were quick and impatient in their approach, and the less skilled users were slower and more deliberate.

A surprising result was that among OSCAR2 users, the three most experienced users (which, given the correlation just discussed, also means the three users who required the least help) gave the lowest SUS scores (see Figure 6-21), whereas for OSCAR1 the relationship was exactly inverted: the most experienced (and least helped) users gave the highest SUS scores (see Figure 6-20). Since in both cases the users had completed the tasks with little or no help, it did not appear to be the case that any of them had struggled with OSCAR to complete the tasks. We are not sure what interpretation to make, but perhaps the more experienced users perceived OSCAR1 as more powerful and/or challenging and actually *preferred* it even though for most people it was more difficult to use.

The most important result of the study is that the usability of OSCAR2 was acceptable because, in general, users were able to create and control connections as well as invoke, create, and edit reusable configurations that made the creation of those connections easier. Nevertheless, the interface is still in need of improvements before it can provide a wide variety of people with the full range of functions of which OSCAR is capable.

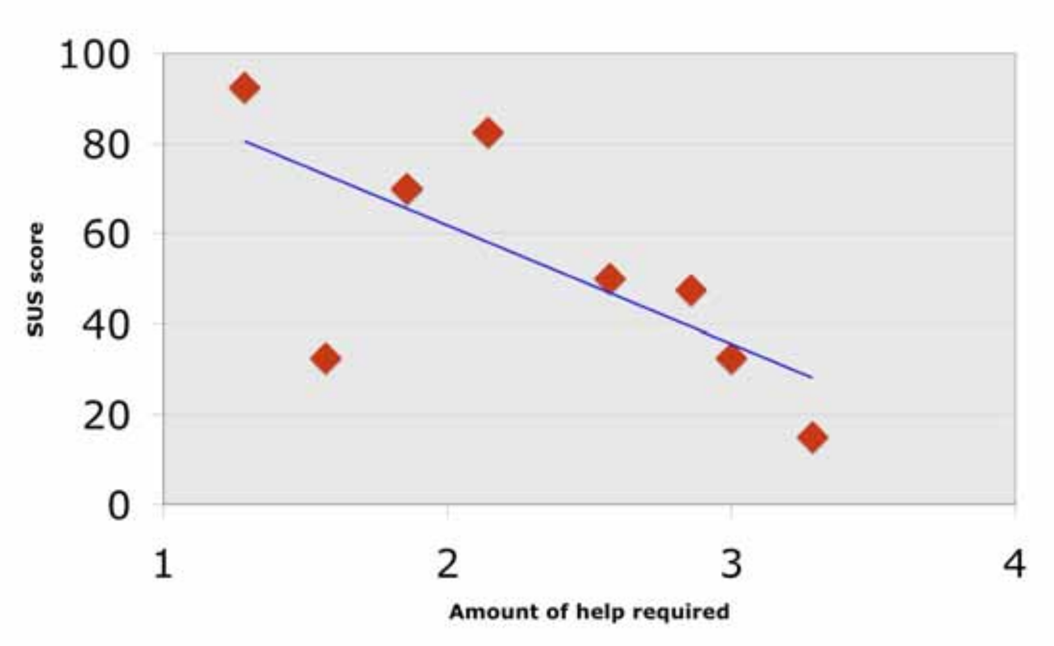


Figure 6-20: The SUS scores given by OSCAR1 participants (shown on the Y-axis) plotted against the amount of help required by each participant (shown on the X-axis). As expected, users that required more help gave OSCAR a lower score in terms of perceived usability.

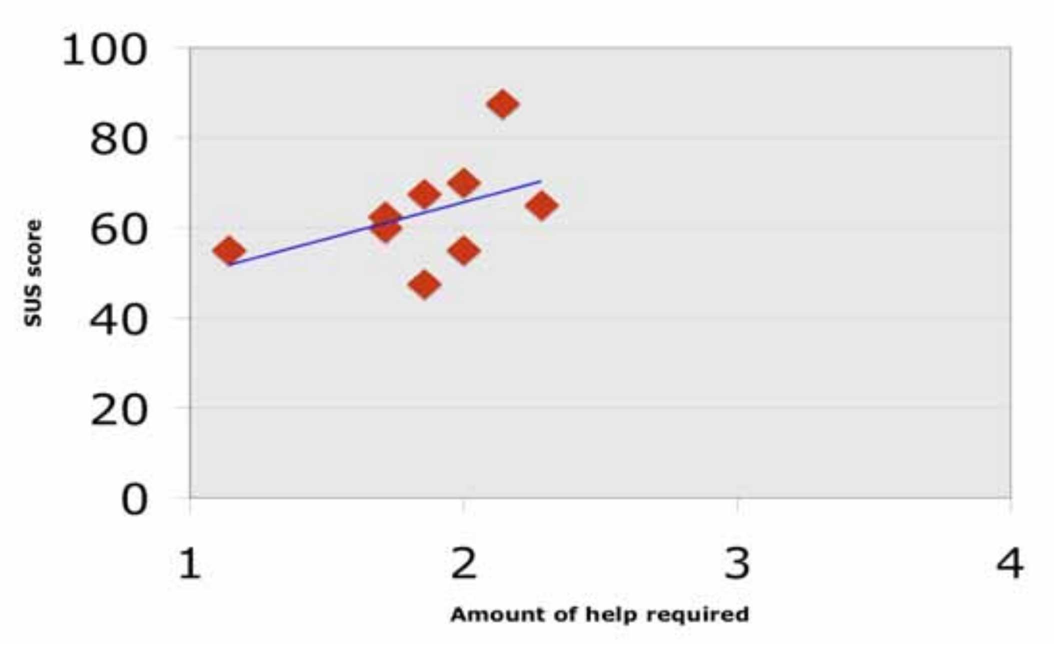


Figure 6-21: The SUS scores given by OSCAR2 participants (shown on the Y-axis) plotted against the amount of help required by each participant (shown on the X-axis). Unlike the relationship observed in OSCAR1 (Figure 6-20), users that required *less* help gave OSCAR a lower score in terms of perceived usability, indicating that users who required less help had a lower opinion of OSCAR's usability.

Perhaps the best indication of users’ reactions to OSCAR, however, came in their responses to the post-test question “Who do you think is this system for?” These representative quotes, shown in Table 6-8, show the improvement from OSCAR 1 to OSCAR 2, as well as the ultimate degree of satisfaction indicated by users to OSCAR 2.

OSCAR 1	OSCAR 2
*“I know several **high-tech types** who’d like it.”* *“Not for everyone. A **normal person** might have a **problem.**”* *“Tech savvy person. **Sharper Image** types”* *“People like me, **who like gadgets** and like to fiddle with things.”*	*“Average person. The **vast majority**. More than 50% of the Bay Area.”* *“**I’m not tech savvy** but I would totally use this. It’s for the average person.”* *“Anyone. Nowadays **we all have lots of devices** and it should be easy to connect them.”*

Table 6-8: Responses to the question “Who is this system for?” indicated a significant improvement between OSCAR 1 and OSCAR 2, and indicated that OSCAR 2 was well received by users.

6.6 Interview Results

6.6.1 Projected Activities

Users revealed a good deal of variety in the activities they would like to carry out with a system like OSCAR. We asked people to indicate the activities they could imagine using OSCAR for in two different ways. First we asked them to simply name the activities they would imagine themselves using OSCAR for. Later we asked each participant to read two lists of possible recipes/setups and rank them according to the usefulness that they, personally, would attribute to each item.

6.6.1.1 Self-described activities

We coded the responses according to the language used by each participant and attempted to collapse similarly described activities into single categories (see Figure 6-22). Some categories ended up being quite broad, such as "Listen to music," and this included responses such as "I'd use it to listen to music," and "music in every room of the house," and "listening to music in the kitchen and the living room," while others were quite narrow, such as "drunk dialing video call." Note that three of the top 5 activities shown in Figure 6-22 closely resembled activities that the users had just performed as part of our task assessment, so it is natural that these would be over-represented in the responses. Thus, while most users mentioned "listen to music" as one of the things they'd like to do with

OSCAR, what is most interesting to us is the wide diversity of creative, unscripted responses that users gave us. This indicates to us that users have diverse and even unique desires in how they would like to assemble, set up, and experience their home networks, and this, in turn, points to the need for frameworks and end-user tools that support end-user composition.

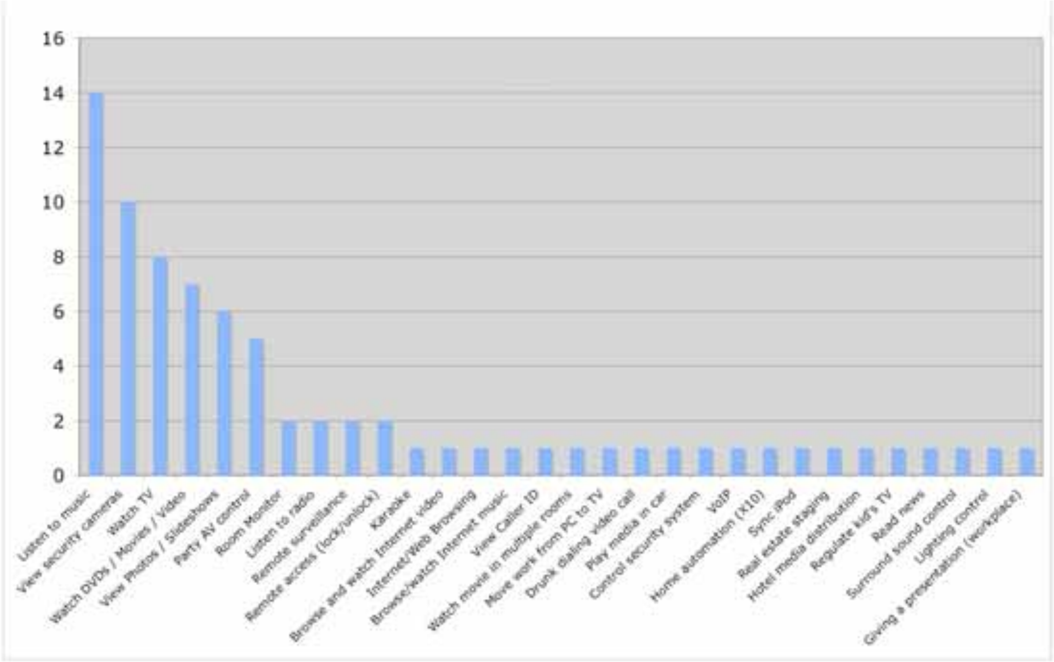


Figure 6-22: When asked to respond to an open-ended question about what they would use OSCAR for in their own households, users gave a range of different responses. Many participants mentioned some variant of the activities represented in the tasks they had just performed (e.g., listen to music, view security cameras, and view photos), but participants went beyond the capabilities that had been demonstrated and came up with several creative uses that had not been anticipated such as "drunk driving video call." Three of the top 5 most-mentioned were activities that the users had just performed: "Listen to music," View security camera," and "View Photos / Slideshows." This graph shows the number of participants (along the Y axis) who mentioned each type of activity (along the X axis).

6.6.1.2 Activity Rankings

The results of the two sets of recipe/setup rankings reinforce the results of the self-described activities to paint a picture of highly diverse needs and desires among potential users of a system like OSCAR and by extension, among users of networked home media in general (Figure 6-23 and Figure 6-24). While there are a couple of notable favorites, the responses are fairly evenly distributed for both the in-home setups (i.e., setups that define compositions of devices entirely within one home) and for setups that involve devices that are shared publicly or between homes.

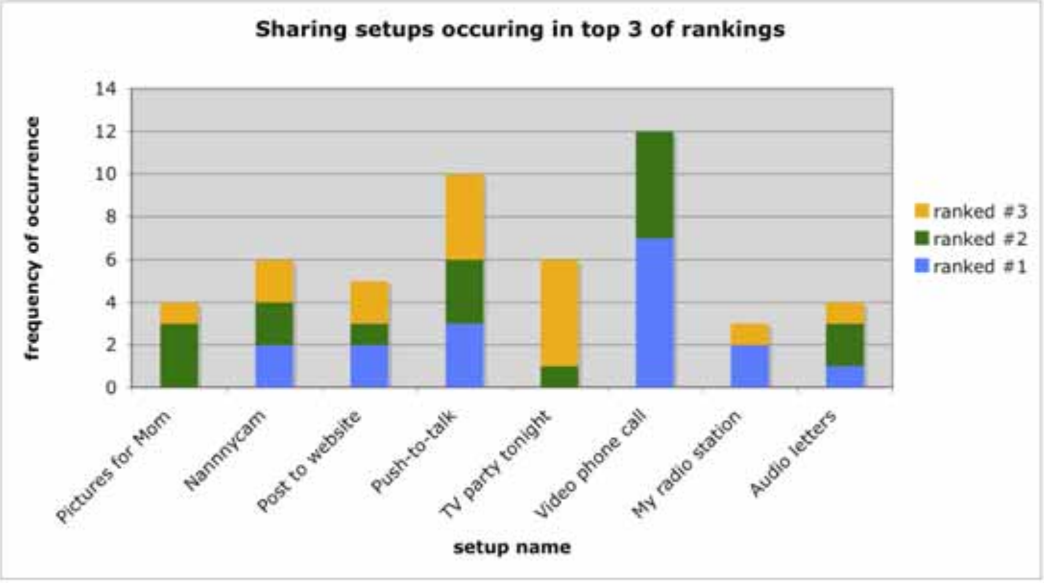


Figure 6-23: OSCAR study participants ranked several potential recipes highly, with six of the nine candidates ranked in the top 3 by at least 1/3 of the participants. Even the relatively unpopular "Room monitor" setup was given a #1 ranking by 3 of the 18 participants, indicating that users' interests in terms of how they would use a system like OSCAR are quite varied. Users' explanations about their choices reinforced the diversity of their interests and also the strength of their interest in performing the activities that OSCAR would make available to them.

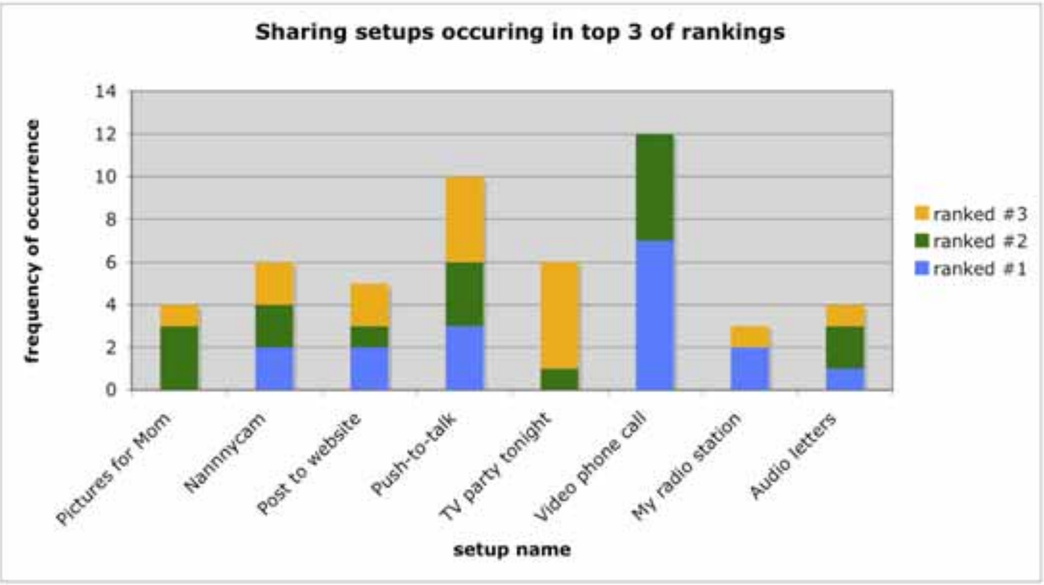


Figure 6-24: Just as different users had expressed a preference for a range of different in-home setups, different users responded differently to the ranking of Setups involving devices and services that were shared across homes. These responses, and the explanations given, again showed the diversity of users' interests.

The reasons that people gave for their responses regarding the activities they could see themselves carrying out with OSCAR helps to shed some light on the heterogeneous needs and desires of home network users.

For example, one respondent explained that she lives with her mother who has dementia. She is very concerned with keeping tabs on her mother's activities and ensuring that she is out of danger, and would welcome any solution that would make it easier for her to maintain awareness of the activities going on in a different room without having to physically relocate. She has no such solution at the present time. Also, she would like to be able to listen to music in different rooms while cleaning without having to carry around a portable stereo as she does now.

Another respondent was interested in the "audio letters" setup because her husband is a long-haul truck driver and she would like to be able to record longer messages for him to listen to while on the road. Currently they communicate by phone and by email (he picks up wifi on his laptop sporadically from roadside cafes and hotels). She also would be able to record TV shows on one central location in the house and be able to watch them from different rooms.

Several of the younger participants immediately responded that they would love to use a system like OSCAR for controlling music, video, and photos in different rooms when hosting a party. One slightly older participant said that she would use it for playing background music when hosting a dinner party.

A few users said that they watch a lot of TV and are frustrated by the limitations of their current setups. They would be willing to put a second display in their living rooms if there were a good integrated way to control both TVs so that, for example, the primary show appears on the larger screen and has its audio routed to the main speakers while simultaneously they are monitoring another show on the smaller screen to watch for important events or simply to wait for the end of a commercial break.

6.6.2 Sharing Attitudes

We asked participants to share their attitudes about the value of a system like OSCAR that allows you to share media and devices not just within your own house but also among the houses of friends and family. We were expecting that the responses would be emphatically positive—that our participants would light up with excitement and say that being able to share devices and media across homes would make this a must-have item. This was not the case. As shown in Figure 6-25, while the responses tilted somewhat towards the positive end of the scale, the overall response was quite varied. Some users felt that this was simply not useful—they couldn't think of anything they particularly wanted to share with others, while others were concerned that this would increase the complexity of the system too much and make it less appealing. Still others were concerned that such a capability might introduce security and privacy risks, reasoning that if I can get out that means someone else can get in. For these people, the potential risks outweighed the potential benefits.

Similarly, we thought that the ability to share recipes/setups with friends, family, and the public at large, as well as the ability to receive such setups, would be clearly beneficial to our users. This capability was even less enthusiastically received than the previous ones, as shown in Figure 6-26. The reasons for negative ratings here were interesting—several of the naysayers rejected the proposition on the basis that it was so easy to create the recipes/setups that it would just be easier to have everyone do it themselves. If they needed help, a more knowledgeable person could talk them through it on the phone. Some other people were suspicious of sharing and running downloaded setups because it might introduce the risk of viruses or something similar.

Note that for both questions, there were quite a few respondents who were positive about the ideas. Especially with regards to the device & media sharing concept, some of the participants were quite enthusiastic. This further reinforces the point that different people will want different things out of a system like OSCAR.

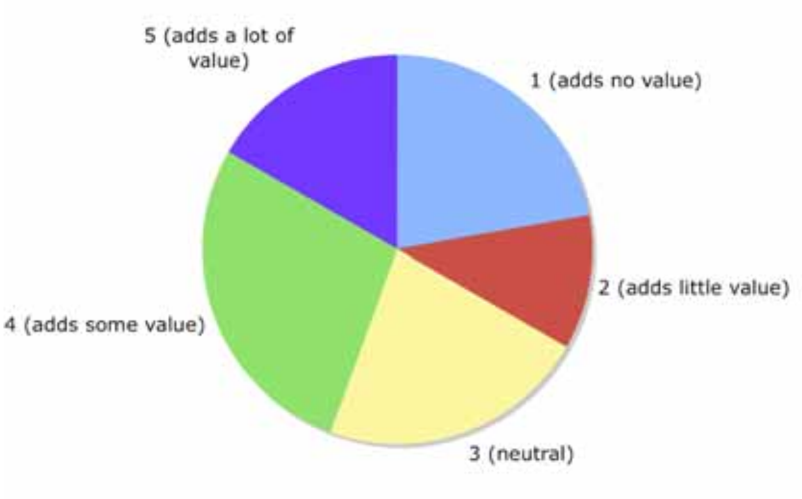


Figure 6-25: Users' attitudes towards the value of a system that facilitates the sharing of devices and media across homes was fairly evenly spread across the spectrum. Even more interestingly, some users were passionately arguing for the importance of adding sharing capabilities to OSCAR while others were just as passionate in denouncing the potential for security and privacy risks and the fact that such a capability would cause them to reject the system outright.

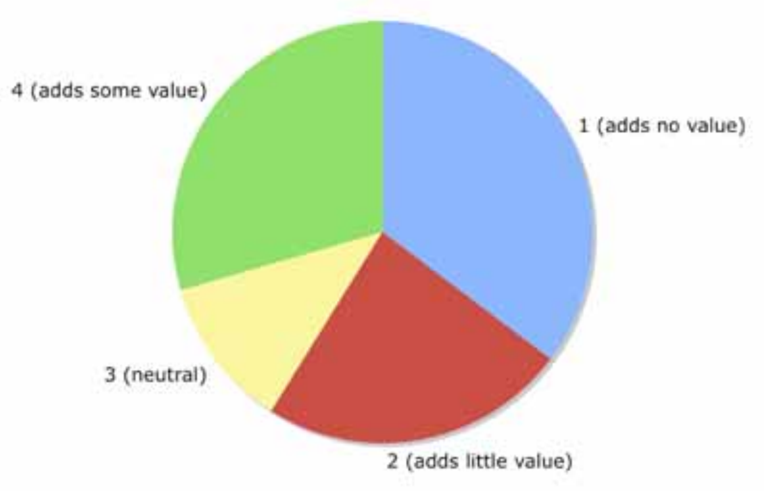


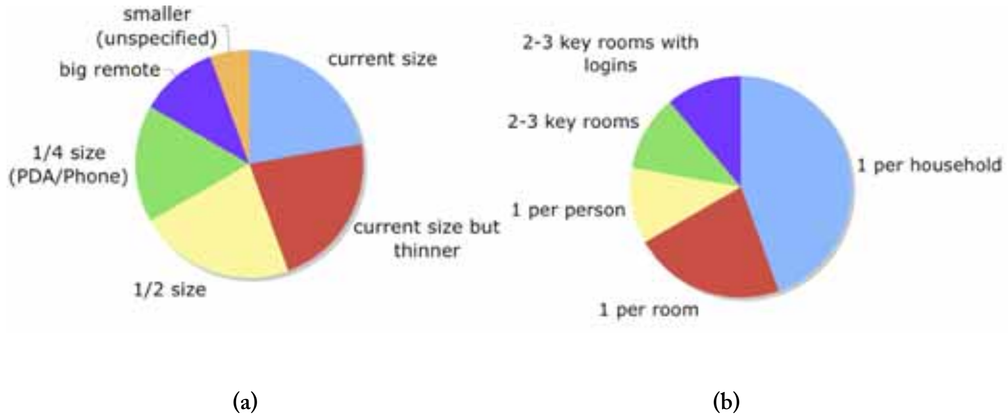
Figure 6-26: Users' attitudes about the value of sharing setups/recipes across households was similarly varied, but somewhat more neutral than attitudes towards shared devices and media. Additionally, there was a larger contingent of people who were strongly negative. One reason for this lack of enthusiasm may have been the sentiment expressed by several users that setups were so simple and easy to understand that sharing them was unnecessary. More complex recipes may have made the motivation for sharing more clear.

6.6.3 Form Factor

Responses to questions about the desired form factor for OSCAR devices and their distribution among locations and occupants of participants' home varied as well, as shown in Figure 6-27. Four respondents said the device was exactly the right size. Another four said the dimensions were right, but it needed to be thinner and lighter. The remaining ten wanted it to be smaller, but how much smaller varied: four said ½ the size, three said ¼ the size (or the size of a PDA or phone), and two said it should be the size and dimensions of a "big remote." Another simply said it needed to be smaller but didn't suggest a particular size.

Eight of the respondents thought that one per household should be enough. Another eight would like to have one per room, or at least one in each of several key rooms (e.g., living room, kitchen, bedroom), and that these would be owned equally by each member of the household. Two of these respondents insisted that the devices needed to have personalized logins, though, so that each family member could get access to their own favorite lists of devices and setups. Two respondents went further down this track and said they would have one device per person instead, and each person would carry theirs around with them.

One person said that the remote would stay in one place—probably by the living room couch. Three others suggested that it should be able to be hung on the wall when not in use—one of these three further suggested that when not in use it could double as a picture frame or other display screen. One person suggested that the functionality be embedded in his phone so he would only have to carry one device. Another suggested that the functionality be embedded in other appliances she already has, like the refrigerator, TV, or thermostat control.



(a) (b)

Figure 6-27: Attitudes about the desired form factor (a) and the number and distribution of OSCAR devices (b) were varied, indicating that different users would prefer to integrate OSCAR's functionality into their homes in different ways.

6.7 Discussion

As stated before, the goals in building and evaluating OSCAR were twofold: to gain insight into the user experience claims of Obje and to design a superior interface for composition and control in home media networks. I believe that OSCAR was able to meet both goals.

The OSCAR interface was designed to elevate connections between devices and media to be primary user interface constructs. Participants understood this concept and were able to make, break, and control connections with no training and very little learning time. Respondents' comments about the devices showed they understood the importance of connections, with one person describing devices as “what you have available to hook up,” and another explaining OSCAR as “a way to have devices interact with each other in a simple and easy to use way.”

Most participants in the study succeeded in using OSCAR to make reusable configurations, but they usually made errors along the way. Even though the reusable configuration might not have been completely correct in meeting a task goal, the users were able to recover from their mistakes and achieve the results the expected. As we will discuss shortly, there remains an open question regarding the extent to which the OSCAR user interface is to blame for the observed errors, as we believe the study design did not properly motivate users to make accurate configurations.

Two key lessons can be gleaned from the improvements in usability and user acceptance from OSCAR1 to OSCAR2 regarding the presentation of reusable configurations to users in this domain: (1) users expect “technical” language to describe challenging concepts regarding technology and (2) designers should prioritize device discovery and control, along with the ability to make one-off, immediate connections over the ability to make reusable compositions. In the OSCAR context, these lessons were learned by observing the improvement that was achieved by switching from the “recipe” metaphor to the “setup” language, and by switching from the activity-oriented interface model to the device/media-oriented model.

The results of the interview indicate the need for an end-user composition system such as OSCAR that provides users integrated, customizable access to their devices' functionality through a highly interoperable, flexible platform such as Obje. The variety of responses to such questions as the activities they would use OSCAR to conduct, the desirability of sharing devices, media, and configurations across households, and the number, ownership, and form factor of client devices indicate the desire of users to configure and interact with their environments differently. As discussed in earlier chapters, support for such varied applications that provide access to the heterogeneous and growing collections of devices that are already present in homes will require that users be given the means to effect their own compositions, rather than relying on software developers to create all of the appropriate applications.

6.7.1 Contributions of OSCAR

OSCAR serves as an embodiment of the ultimate overarching goal of this dissertation's research: to provide an integrated user experience of interacting with multiple networked devices and services. Firstly, OSCAR serves as a reification of the capabilities of Obje, in that it provides users with the ability to browse and discover devices that have been located via ad hoc discovery, control those devices individually, and create, monitor, and control connections among those devices. Additionally, OSCAR supports reusable compositions (called "recipes" or "setups") that encapsulate common activities. These compositions are created, edited, and invoked by end-users. This collection of functionality is unique to OSCAR, and represents a central contribution of this dissertation.

The ability to browse and select arbitrary devices is important because it reduces or eliminates the amount of up-front configuration that is required for devices to interact with each other and to be controlled by OSCAR. While it remains true that devices will have to be provisioned and configured to be made available on the network (or, more generally, on one of the networks that is accessible to OSCAR whether via its primary network interface or via an intermediate *Aggregate*/proxy service), in most cases such configuration will be required for such devices to satisfy their core functionality as networked devices. No special or additional configuration will be required in order to work with Obje/OSCAR. Most importantly, once a baseline configuration of OSCAR and a set of Obje services has been established, the addition of new devices can be accomplished with no additional effort on the part of OSCAR or any of the existing devices. The ability to browse discovered devices is not unique to Obje/OSCAR, as a client written against any platform that supports ad hoc discovery (e.g., Jini [158] or UPnP [153]) would provide some version of this ability. Obje's ability to easily bridge alternative discovery protocols via the *Aggregate* mechanism means that it may be more adaptable than alternative approaches to the addition of a wider variety of devices, though this aspect was not tested in the OSCAR testbed or user study.

OSCAR's ability to control arbitrary devices without explicit foreknowledge of them is another important component of its ability to provide an integrated user experience. Obje services provide their own, customized user interfaces via mobile code to OSCAR, and OSCAR is able to display such UIs on the screen without needing to understand the semantics of their contents or the means by which the UI code communicates with its backend services. Thus, again, new types of devices can be added to the network monitored by OSCAR and these new devices can be used alongside others with no additional configuration. The ability to provide opaque access to arbitrary device control user interfaces is, again, not entirely unique to OSCAR, as the Jini ServiceUI project [156] endeavored to provide similar functionality. Platforms that allow devices to provide web page URLs (e.g., Cooltown [83] and UPnP [153]) could be used to deliver control UIs to clients, though additional specification of the contents and functionality of the content provided at the other end of the provided URL would be required to adapt such a platform for this use. In all cases, the functionality of alternative platforms (Jini, UPnP, and Cooltown) does not include OSCAR's ability to receive and display user interfaces that have been "pushed" by the services.

OSCAR's most significant contributions become clear when we begin to focus on the ability to connect devices, monitor and control multiple ongoing connections, and display control user interfaces for each of the devices involved in the transfer. Being able to create and control connections allows users to gain the benefits of the high degree of interoperability provided by Obje.

OSCAR presents a generic user interface that gives users access to a wide range of devices and, through the interaction sequences, allows users to discover what devices can be connected with which other devices. In addition to delivering a user experience of rich interoperability, OSCAR provides an integrated user interface for interacting with all of one's devices and services within a home network. Users do not need to learn different techniques for establishing, monitoring, or controlling connections.

Since each Obje component can provide its own custom control user interface, there is no guarantee that, in an open system where different components are produced by different sources, each user interface will operate in a similar way. This means that the integrated user experience may be compromised somewhat by the lack of control user interface similarity. However, since OSCAR does provide the ability to collect and display multiple user interfaces in an integrated control panel, at least all controls can be accessed in one convenient place. Additionally, it was not the goal of either the Obje or OSCAR projects to dictate the specifics of how components are controlled or how they should construct their user interfaces. As these concerns are orthogonal to the research in the dissertation, it would be possible to incorporate the results of a research project like Nichols, et al.'s Uniform [115] to ensure that users could have consistent control user interfaces across multiple devices. All that would be required is to dictate that components provide their control specifications in a standard format (e.g., the Pebbles Universal Control specification [114] supported by Uniform) rather than leaving the format of the user interface unspecified as Obje and OSCAR do.

Finally, OSCAR provides users the ability to create reusable and sharable (within a single household) compositions that encapsulate commonly-executed media-oriented activities that are tagged with user-generated names for later recall and access. This ability is again built atop Obje, but is not a direct mapping of one of Obje's features. The need for reusable compositions was demonstrated by earlier studies of home technology use by myself and others [55, 128, 129], which showed that such compositions will help support both the temporal and social division of labor that are essential to the smooth functioning of households. OSCAR's compositions, ultimately called "Setups," allow users to specify connections among source and destination devices both concretely (by specifying the exact devices to connect) or abstractly (by describing the criteria for selecting devices), and the rules for how the source(s) and destination(s) should be connected when the setup is invoked. This design allows for a variety of types of connections to be specified, giving users flexible and customizable control over their networks of devices.

6.7.2 Significance of the OSCAR User Study

During its design, OSCAR was iteratively prototyped using multiple methods, and was subjected to two rounds of expert evaluation. The primary contribution of these studies is to show that the contributions claimed by OSCAR can be realized by end-users of varying technical background. That is to say, even non-technical users were able to use OSCAR to browse, select, connect, and control devices, and they were able to create, edit, and invoke reusable compositions of devices.

In addition, potential users reacted positively to the idea of using a system like OSCAR in their own homes. Based on the interviews, it was clear that these reactions were based on a number of factors, each of which were weighted differently by different users. In particular, users appeared to react positively to the following features of OSCAR and Obje:

- the notion of easy connections among devices in different rooms—especially routing audio and video anywhere (e.g., listening to the same music collections in the living room and kitchen)
- reusing devices for multiple purposes (e.g., using the living room TV to view photos and the front door security camera)
- accessing multiple disparate functions from a single control point (e.g., using OSCAR to play music, select photos, and see who's at the front door)

- the ability to create presets for easy activation of media activities (e.g., create a Setup to easily play specific music in the kitchen)

The study also highlighted some design issues for designing an end-user composition system for home media. Firstly, based on the different reactions and task success rates between OSCAR1 and OSCAR2, it is clear that the use of neutral, technical-sounding terminology for describing reusable compositions (such as "Setup") is more desirable, easier to learn, and easier to use than terminology that is apparently friendlier but perhaps loaded with alternative associations (such as "Recipe"). While it is clear on the one hand that the particular term "Recipe" and it's attendant terminology such as "Ingredient" and "Prepare" were seen as especially poor fits for OSCAR's particular domain, it is also clear from users' reactions that a more technical set of terminology would be more communicative and less dissonant than any terminology derived from less technical aspects of, for example, domestic life.

Secondly, for the domain of home media control in particular, presenting users first with a user interface for browsing and selecting among their various devices and media is preferable to and less confusing than presenting them first with a list of pre-defined configurations. In part, this reaction may have been based on the fact that all users received their first exposure not only to OSCAR, but to all of the devices in their "house" during the study, and were not given the opportunity to have a repeat experience after becoming familiar with either aspect of the simulated environment. Nevertheless, it makes sense that upon a first experience with an application like OSCAR, it would be beneficial from the perspective of building users' understanding and trust for the application to first communicate its basic capabilities—i.e., that it can discover and provide information about a particular set of devices in the home—before revealing more advanced features such as the ability to create and invoke reusable compositions.

The interview component of the OSCAR study also highlighted the varied needs and preferences of users with respect to a home media control environment, and this variety supports the need for composable services, delivered through a framework that supports robust interoperability, that can be flexibly customized by end-users to meet their particular needs. In particular, users indicated a wide range of different media-related activities for which they would use a system like OSCAR/Obje, and they also indicated a range of different attitudes towards the sharing of devices and media across households, the sharing of compositions (Recipes/Setups), and the form factor and ownership of control devices. These responses indicate that it would be considerably costly and inefficient to support the variety of user needs via specific, developer-built applications and devices.

6.8 Summary

In this chapter, I described a two-phase user study (including an inter-phase redesign) of OSCAR—and end-user composition tool for home media networks that builds upon Obje's user experience goals and extends them by allowing the construction of reusable compositions for common tasks. In so doing, I also described the final phase of OSCAR's design and development, allowing us to see this project's final embodiment of a system that supports an integrated user experience of interacting with multiple networked devices and services.

A user study with 18 users with varied backgrounds showed that people could use OSCAR to configure and control a realistic and fully operational home media network, but that they made a number of errors constructing reusable configurations that would prevent them from having an optimal experience of their networks over a longer period of time. Users gave OSCAR positive subjective assessments in terms of perceived usability and desire to use such an application in their homes.

The study presented in this chapter shows that users can perform composition using a tool like OSCAR, and that they wish to be given the means to do so. This a key result of this dissertation, and represents a positive evaluation of OSCAR itself, but also of the basic user experience goals of Obje and the possibility of end-user composition more broadly. From this vantage point, it is now possible to look forward and ask what can be done to further improve the capabilities and usability of end-user composition systems, and what lessons have been learned from my work that can be applied to the wider class of systems that will allow end-users to take control of their emerging networks of devices. In the next chapter, I will discuss a number of directions for further research on these topics.

7 Future Work

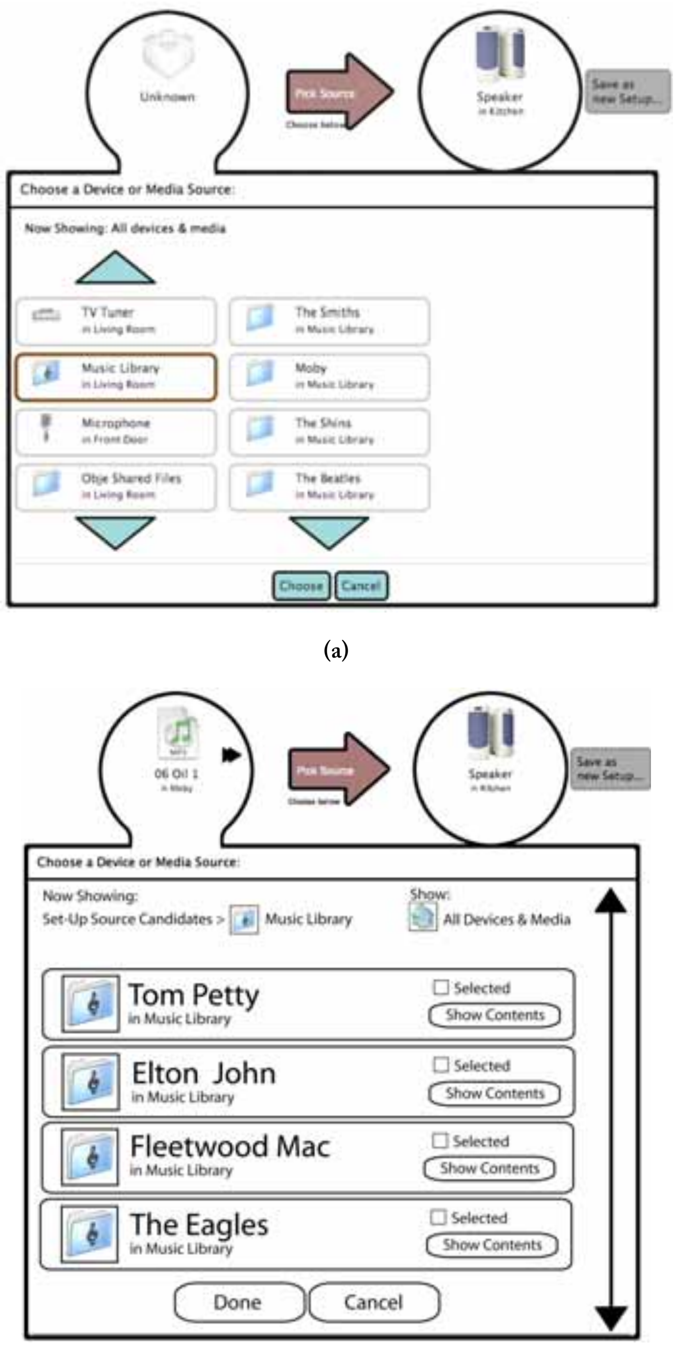
The OSCAR user study showed that users of varying technical background could make effective use of OSCAR to browse, control, and connect devices and services in a home media network. Returning now to the higher level goal of delivering an integrated user experience of interacting with device-rich environments, I will devote this chapter to a discussion of what remains to be done to further improve the degree of integration experienced by users. I will first discuss changes to OSCAR to improve the usability of the functions it currently supports. I will then discuss changes to OSCAR to enhance the range of compositions that can be created by users, such as support for connections involving multiple sources and destinations simultaneously and context-aware Setups. In order to support sharing of devices and services across local network boundaries, additional mechanisms will need to be built atop Obje, and I will discuss possible mechanisms. Similarly, additional work will need to be done to support sharing of Setups among different households. Finally, I will discuss the great potential benefits of creating and supporting wide area communities of practice via shared Setups, and I will discuss the challenges of constructing and studying those types of communities.

7.1 Improvements to OSCAR

As discussed in the latter part of Chapter 6, even though OSCAR2 represented a vast improvement over its predecessor, there remain areas for further improvement.

7.1.1 Usability Improvements

A number of "garden-variety" usability issues continued to afflict OSCAR2. The touchscreen was occasionally slow to respond to user input, which caused considerable usability problems as some users believed they had made an incorrect selection and went on to take other actions while in fact the system was still busy processing their original correct input. Users had difficulty determining how to select all of the contents of an Aggregate (e.g., all of the songs by the Beatles within the Music Library) as opposed to expanding the Aggregate to select individual contents (see Figure 7-1 for a proposed redesign that addresses this particular issue). These types of usability issues are typical of a wide variety of different applications and, while they can cause serious problems for users, it is not particularly worthwhile to discuss their individual solutions in this document.



(a)

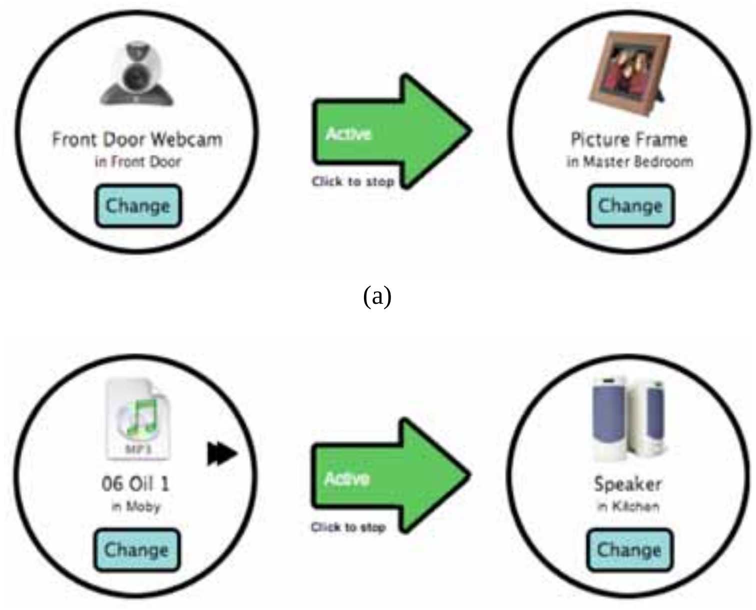
(b)

Figure 7-1: A possible redesign for the OSCAR browser that would allow users to clearly distinguish between selecting an Aggregate component and showing the contents of the Aggregate to select among its contents. In the original design, the user could select an Aggregate by tapping "Choose" when the Aggregate was highlighted (e.g. the "Music Library" as shown in (a)), or could tap on one of the sub-Aggregates (e.g, "The Smiths" or "Moby") that is shown to the right of the selection. In the proposed redesign (shown in (b)), the operations for expanding an Aggregate ("Show Contents") and selecting it are clearly delineated.

More interesting is the discussion of a pair of issues that are fundamental to the approach taken by Obje and OSCAR, and that will require further research to resolve effectively: the tradeoff between generic and domain-specific operations (and the terms used to describe them), and the difficulty of expressing rules and criteria for service selection when creating or editing Setups.

7.1.1.1 Trading off generic and specific operations

As discussed in Chapter 6, one subtask that gave many users problems was removing an image from the Front Door Picture Frame. The expected solution to this was to click the arrow in the *Connection Details* screen that was labeled (albeit in small letters) *Click to Stop* (see Figure 7-2). However, this was not apparent to users as it was not clear that "stopping" a still image would remove it from the frame. Not surprisingly, users were more successful when we asked them to "kill" the song that was playing—the term "stop" is seen as a close enough synonym to "kill" in this context, whereas "stop" does not mean "remove" to most people.



(a)

(b)

Figure 7-2: The language for controlling active connections in OSCAR remains generic, regardless of the types of media and devices being connected and controlled. In the two examples shown in (a) and (b), the label describing how to stop the connection is the same for both a webcam and streaming audio connection. As this caused difficulty for some users, an area of possible future work is to look at how to support more finely tuned language choices for controlling different types of connections without requiring clients such as OSCAR to have extensive advance knowledge of the semantics of different media and device types.

In general, one of the strengths of OSCAR—its lack of detailed knowledge about any particular type of device or media—is also one of its deficits. While OSCAR achieves significant flexibility and adaptability by allowing users to simply "connect" a wide variety of "sources" and "destinations" without regard to whether the connection is between a song and a set of speakers, a webcam and a picture frame, a photo album and a hard drive used for data backup, or, for that matter, a scanner and a printer. Users are more likely, however, to think in device or media specific terms like "playing" a song, "viewing" a web cam, "storing" an album, or "printing" a scanned image. This tension is inherent: as an application becomes more tightly adapted to carrying out specific tasks, it becomes less capable of taking on new or unanticipated tasks.

One compromise solution that may be appropriate for OSCAR in particular is to use media-specific terms for known types (audio, video, image, etc.) and default to generic terms when the specific language cannot readily be determined. In most cases, the correct term for the operation can be determined by the destination device. The appropriate term could either be supplied by the destination services via an expanded service API (e.g., the Obje *DataSink* interface could be expanded to include a method like `String getConnectionOperationTerm()`), or could be determined by a mapping within OSCAR itself that uses heuristics (e.g., if destination data type is "audio/*" then use term "play," if destination data type is "image/*" use term "view") to select the appropriate term. This latter approach may be difficult to implement in practice, however, as it is hard to imagine a set of heuristics that could reliably determine the appropriate use of the term “print” (as opposed to “view”) for a connection involving an image and a printer without requiring OSCAR to know in advance which devices are, in fact, printers.

7.1.1.2 Setups, and Setup Details

OSCAR2’s users continued to struggle with various aspects of creating and editing Setups. Identifying the devices to use was not particularly difficult, but selecting among the options for rules that would be applied at a later time, when the Setup was run, appeared to present difficulties for users. Several users reported that they simply did not read the options, or that they read them once, failed to understand them, and decided not to use them. A common strategy among these users was to simply add all of the devices to the Setup, and then “run” the result to see what happens. Any undesired effects were corrected using the controls available for manipulating live connections, which had the end result of leaving an incorrect, or at least non-optimal, Setup in the list of Setups. On the other hand, some other users were able to use the rule-setting options quite adeptly.

The UI for editing Setups underwent significant changes among the various versions of the OSCAR prototype (see Figure 7-3), and while the final OSCAR2 version appeared to be the most successful from a usability perspective, it still leaves something to be desired. It therefore begs the question: would further iterative changes improve this user interface to the point where a majority of users would be able to quickly master the available options, or is a substantially different approach required? Further research is required to answer this question.

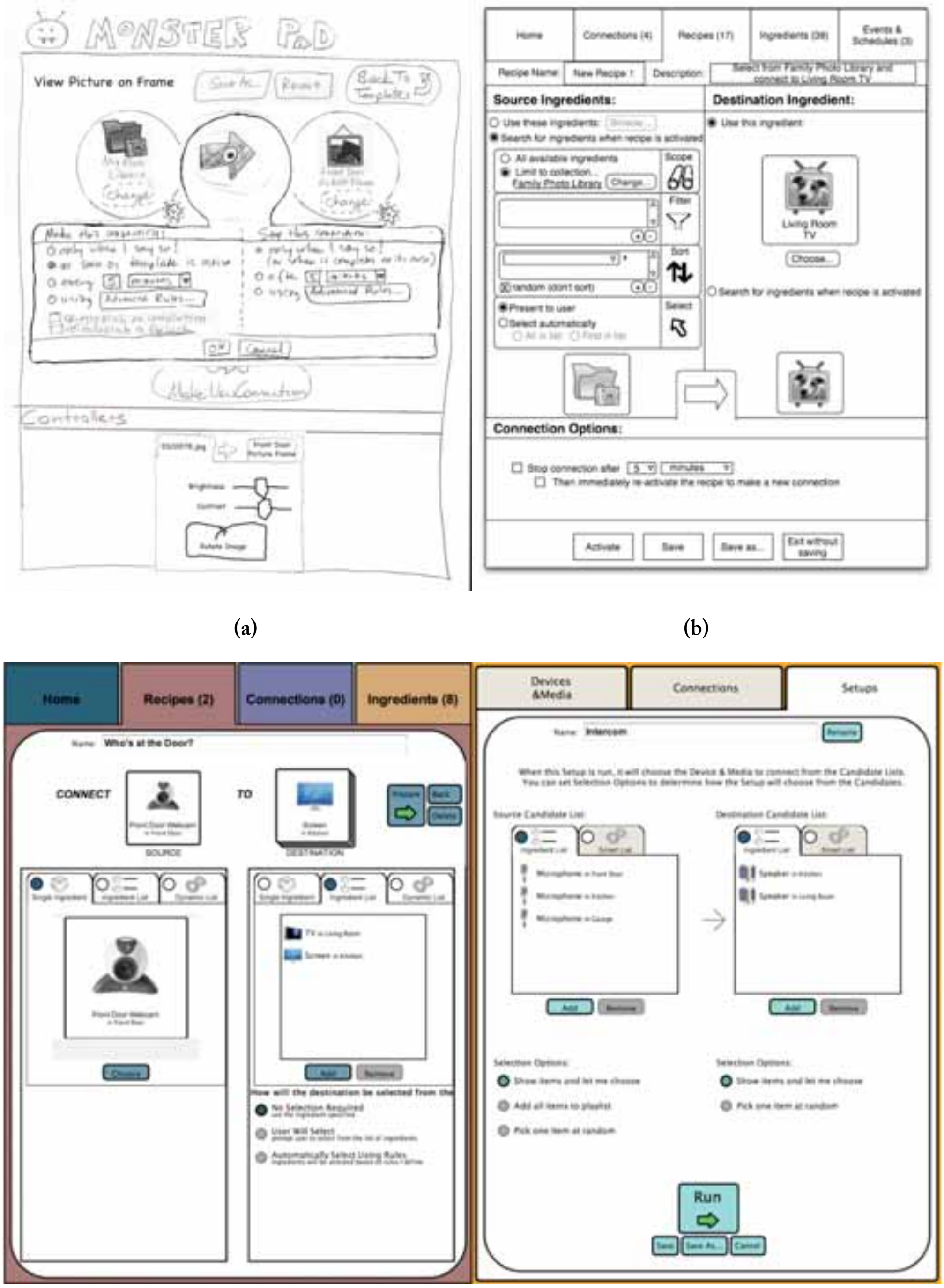


Figure 7-3: The "Setup Details" screen received more attention during design and was subjected to more changes than any other screen in OSCAR. Here the versions produced for the OSCAR Paper Prototype (a), the OSCAR Medium Fidelity Prototype (b), OSCAR1 (c), and OSCAR2 (d) are shown for comparison. Despite the intensive effort, the final version was still difficult for some users to use.

Simple improvements could go a long way. Several users indicated at various points during the study that they would have consulted a manual or help section. Such a function would certainly need to be included in any widely distributed version of OSCAR, though the details of how it is incorporated (e.g., as a separate global tab, as a function available from each individual screen or even each section of each screen, or as a separate printed or online manual) would need to be worked out. Similarly, a simple improvement to the process of constructing Setups such as incorporating "wizard" functionality to guide users through the process in a more structured way could have a substantial impact on usability.

A more ambitious concept would be to downplay or even eliminate the notion of Setups as things that are explicitly and intentionally created by users. If instead, OSCAR simply kept track of all connections made by users, it could potentially learn the common patterns of interaction and automatically generate Setups to facilitate easier re-invocation of those patterns in the future. Alternatively, the history of connections could be presented to the user along with tools to allow them to extract, encapsulate, and selectively generalize useful patterns for later reuse. Such approaches would benefit from techniques that have been developed to support programming by demonstration (e.g., [29] and [86]).

A particularly intriguing direction for such further research would be to empirically compare the option-setting approach taken in OSCAR with other UI approaches that have been developed for similar types of scenarios, such as the purely iconic approach taken in the Jigsaw Editor [70, 127], and the constrained natural language approach taken in CAMP [150]. These two would be particularly apt for comparison, because the Jigsaw Editor represents a visually compelling interface that presents a limited expressive range in that users are limited to creating linear chains of operators whose relationship to each other cannot be parameterized or otherwise controlled (see Figure 7-4), whereas CAMP potentially allows a greater range of expression for users, but is based entirely on textual representations of objects and operations (see Figure 7-5) and may introduce complexity beyond that presented by OSCAR. In other words, OSCAR may be seen as a midpoint between CAMP and the Jigsaw Editor, and a comparison of these approaches as applied to a consistent set of tasks and evaluated by a consistent set of users may provide significant insight. Of course, other user interface approaches could, and perhaps should, be created and evaluated as well.

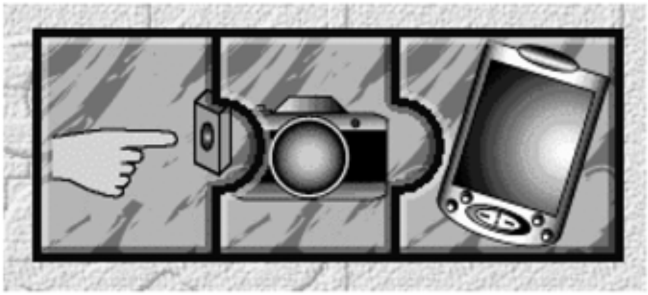
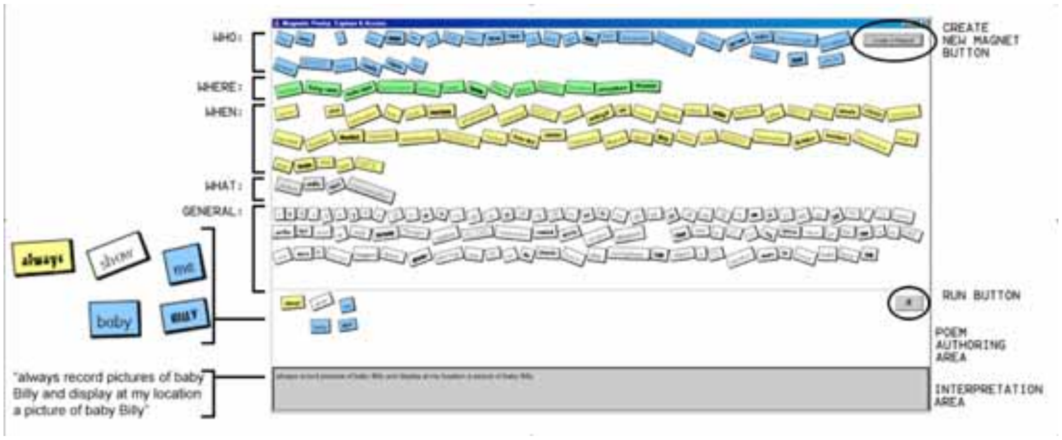


Figure 7-4: The Jigsaw Editor [70, 127] represents an end-user programming user interface that is based entirely on visual representations of objects and linear combinations of them to form processing rules. This user interface may be more immediately compelling for users than OSCAR's option-setting dialogs, but the lack of control over configuration options could lead to eventual frustration.



"always record pictures of baby Billy and display at my location a picture of baby Billy"

Figure 7-5: The CAMP user interface [150] is based on a "magnetic poetry" metaphor. The textual programming interface provides users with more fine grained control than OSCAR, but the range of options may also lead to a reduced ease-of-use and steeper learning curve. Further studies are required to discover the relative strengths and weaknesses of Jigsaw, CAMP, and OSCAR.

Yet another direction for

7.1.2 Increasing OSCAR's range of functionality

An additional area for improvement would be to enhance OSCAR and Obje to support a greater range of functionality. The ability to compose and control data transfer connections among devices and services in a home media network provides a solid foundation upon which to explore further compositional capabilities.

7.1.2.1 Context-Aware Setups

The Obje metadata mechanisms allow clients and services to provide information about their contextual properties such as location, usage history, and ownership. Service properties can be updated dynamically, and the updates are propagated to clients that have registered to be notified of such updates. Similarly, client properties can be updated as contexts change, and clients can react to changes in their own context. In addition, the OSCAR Setup data structure is designed to allow Setups to be designed to make use of contextual data. The smart list feature plays this role. It allows a user to define Setups by defining criteria to select the source or destination devices, rather than defining the specific devices. Thus, for example, a smart list could be used to define a Setup that would select all destinations that are located in the Living Room and accept audio connections. This list could then be presented to the user to select among, or one of the choices could be selected automatically or according to some other ranking criteria. Such a setup would be able to take advantage not only of relatively static devices in the living room, such as the hardwired speakers, but also mobile devices that happen to be in the room at the time, such as laptop speakers or portable music players. In essence, such a Setup can take advantage of the context of whatever devices it has available at the time of its instantiation.

Such an ability to adapt to the current context is only the beginning of what would be useful. A critical addition to these facilities would be to enable *relative* criteria to be defined in terms of the context of other parties in the Setup. For example, rather than defining the above Setup in terms of all destinations in the Living Room, it would be even more powerful to define a Setup that will use whatever devices are *in the same location as the client* when the Setup is invoked. While defining a criterion as "Location = 'Living Room'" enables a class of useful Setups, defining a criterion as "Location = Client.Location" or something similar would enable an even more appropriate set of possibilities. Along these lines, it would be useful to define a set of criteria such as "most recently used," "most frequently used," "nearest to me," or "owned by one of my friends," which would have to be computed dynamically based on the current contexts of the client and potentially multiple components.

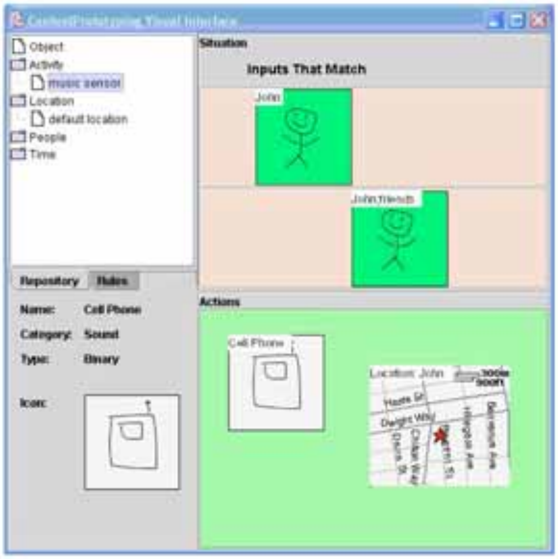
The ability to create and use context-sensitive Setups was not explored as part of the research described in this dissertation, in large part because there does not yet exist an implementation of Obje that includes reliable contextual data such as location. Such an effort has been begun [154], but had not come to fruition at the time of OSCAR testbed development. Rather than attempt to create a reliable system or an effective simulation of such, I focused the OSCAR experiment on connection and control of near-future networked home media devices without assuming the additional benefits of dynamically updated context. Adding additional support for dynamic context, especially support for relative criteria, would increase the functionality of the Setups users could create, but would also doubtless increase the complexity of the options available to users, and so additional design and user evaluations would need to be conducted.

7.1.2.2 Event-triggered Setups

In addition to supporting Setups that dynamically adapt themselves to their surrounding context, it would be of value in many cases to allow Setups to be triggered by changes in context or other events. For example, a user could create a Setup to play the morning radio news in their kitchen that is automatically triggered by their entering the kitchen between 6am and 7am. A variety of parameters could be used to define Setup triggers, including time and day (daily at 7:00am, at 5:00pm Monday through Friday, etc.), change in availability of a device or person (when my iPod becomes available, when Angie enters the house, etc.), and more complex compound criteria (when I arrive home and it's dark outside, when any of the children leave the house, when my iPod becomes available and it has not been connected to the library in more than 24 hours, etc.). Again, adding such capabilities will expand the range of expressive possibilities available to users, but accessing such possibilities will introduce new complexities.

Defining complex criteria and triggers is a major focus of the iCAP system [33], which uses a sketch-based graphical layout to allow users to create IF-THEN rules where the satisfaction of the criteria specified by the "IF" clause will cause the actions specified by the "THEN" clause to fire (see Figure 7-6). Whereas OSCAR has focused largely on allowing users to define complex actions to take (e.g., connect all of the songs by the Beatles to a set of speakers that I will select when I invoke the Setup), and has done little to support fine-grained control over the conditions that would cause such actions to fire (OSCAR essentially defines a single immutable "IF" clause: If I invoke the Setup manually), iCAP has focused largely on allowing users to exercise great control over the conditions under which actions should occur, and does not claim to allow detailed control over the actions themselves. Combining the strengths of iCAP and OSCAR, therefore, would be a promising direction for further research.

Both OSCAR and iCAP have been shown through user studies to be capable of effective use by non-technical end users, yet the combination of both approaches would lead to an increase in complexity that would require further studies with end-users.



Name: Cell Phone
Category: Sound
Type: Binary
Icon:

Figure 7-6: iCAP's sketch-based user interface [33] provides users with a great deal of control over the conditions under which context-triggered actions should fire. However, the control over the actions themselves is less of a focus of the work. Combining iCAP's strengths for rule-setting and OSCAR's strengths in defining complex actions could provide users with a high degree of expressiveness and control. Further research is required to determine whether such increased power can be provided without sacrificing usability.

7.2 Studying OSCAR in daily use

A number of issues relating to OSCAR's usefulness and usability would be likely to emerge in a long-term deployment. As discussed in Chapter 6, it may be that users will eventually correct errors in the Setups they create after they see the effects of running those Setups repeatedly.

Another issue that may arise in daily use is the level of complexity that could arise from running multiple Setups at the same time that attempt to make use of the same devices. Given the relatively small size of users' current and near-future home environments (perhaps O(10) different devices being connected together), such situations may be easy to identify and resolve—whether automatically or with the assistance of the user. However, as networks grow in size and functionality, resource contention issues are likely to become more and more severe. In addition, they will become more challenging to resolve when multiple household members are permitted to create and run Setups independently, and the ability to monitor and control such Setups is distributed across different client devices.

A third issue that would likely be illuminated by regular use is the need for support for error recovery and troubleshooting. These issues are especially challenging because errors can arise at a variety of levels, some of which may not be detectable by Obje or OSCAR. For example, while Obje/OSCAR can detect and report issues arising from data type mismatches, single-device failures, or software bugs that cause premature termination of a connection, it cannot easily diagnose failures of the entire network (e.g., home router failure) or hardware problems such as power failures. Providing support for such general troubleshooting presents a significant research challenge in itself.

7.3 Sharing Devices and Media

As part of the OSCAR user study, I interviewed users about their interest in the ability to share devices and media across households. I also asked them about their interest in sharing Setups with others outside their homes. In both cases, the reaction was mixed, with reactions ranging from strongly positive to strongly negative. Thus, while such functionality is not necessarily appealing to all users, it is extremely appealing to some of them, and it would be a valuable area for future investigation to understand how to support such features.

Sharing certain types of media across homes is already possible to some extent using web-based services such as Flickr [167], YouTube [169], and SHOUTCast [117]. These systems allow one, respectively, to share personal photos, personal videos, and create a streaming radio station from one's own music library. These shared media can be made available to anyone on the web, or can be made available to only a subset of friends, family, or other authorized users. Note that these technologies support somewhat limited usage scenarios, and require some degree of technical sophistication to create, configure, and access the collections. Due to licensing restrictions, they do not easily permit, for example, one user to share a commercial video or recorded TV show with a distant family member, even though such limited sharing may be legally permissible under some circumstances. They do not allow a remote friend to peruse my private music collection to see what new artists I have been listening to in order to get ideas for their own future purchases.

Sharing devices across households is even more challenging and rare, though not unheard of. CEIVA digital picture frames can connect to a proprietary service that allows friends and family to post photos to a web collection that in turn feeds images to a user's home picture frame [20], effectively allowing users to "post" images to a remote picture frame. Effecting similar control over devices in others' households, or over one's own devices from points outside the home, would allow for more seamless communication and sharing over a distance. For example, ambient video and audio could be used between selected rooms in two households to allow members of the two households to feel connected to one another despite the distance. Remote access to security cameras may increase the peace of mind of a traveling homeowner or parent. The ability to invoke brief and spontaneous audio or video connections with distant loved ones allows a greater range of communication options to help maintain and strengthen relationships.

While most of the mechanisms provided by Obje and OSCAR for device and media composition are not restricted to local communication, there are additional issues that emerge when considering non-local communication that have not been a focus of OSCAR's or Obje's development. As discussed earlier, Obje uses Zeroconf [72] for its bootstrap discovery. This means that an Obje client will be, by default, restricted to discovering services on the same subnet. Such restrictions provide convenient constraints in terms of security and access control that need to be reexamined when designing for wide-area sharing.

Sharing across networks in Obje/OSCAR is not difficult. Obje employs Zeroconf's local-link multicast discovery to distribute component URIs to clients. Supporting alternative methods for providing such addresses is all that is needed to expand sharing beyond the local subnet. URIs could be transmitted using an out-of-band mechanism such as email, or could be registered and discovered using fixed directory-based services resembling UDDI [152].

There will, of course, still be issues about access control and security—if a URI is all that is needed to access someone's device, there is a danger that such URIs can be obtained by eavesdropping or even guessing. To prevent the unauthorized use of a URI to access a service, secure credentials such as certificates must be used in conjunction with service access. Certificate distribution and management can be challenging in highly distributed systems such as the networks supported by OSCAR and Obje. One solution to credential distribution is to employ the approach taken in Casca [41], which is an application of the more general location-limited channel approach described in [7, 136]. In these approaches, two parties exchange initial credential information over a privileged channel that is relatively immune to eavesdropping such as infrared or physical coupling. The primary drawback to these approaches is that they require that devices are in proximity at some point, which presents a problem for connecting devices that are in distant locations. Alternatively, a centralized server can be used to ensure that initial credentials are exchanged securely. After initial credentials are exchanged, the Casca experience shows how secure sharing can be extended to other devices under the control of either party, thus allowing sharing permissions to be established at the user- or local network-level rather than the individual device level.

Even after the basic mechanism for enforcing security and access control are in place, however, there will still be the need to allow users to describe enforceable policies to control how, when, and by whom their devices can be accessed. Such considerations require work at both the infrastructure level and the end-user level. Temporal role-based access control frameworks have been investigated [77] and may provide a fruitful avenue for exploration in the present domain. Supporting end-users' ability to reason about and express coherent policies remains an open area of research although some recent work has made progress (e.g., [157]).

7.4 Sharing Setups Across Households

Being able to share setups across households is important primarily because it supports the division of labor across a wider unit of analysis. In the study reported in [55], some householders reported that they occasionally provide system configuration assistance for friends and family outside the home. Others in the same study stated they had on occasion received such help from others. These results were corroborated by the interview responses received during the OSCAR user studies.

However, enabling such functionality comes with challenges. For a Setup created in one household (Household A) to be of use in another household

(Household B), all of the information about the devices addressed by the Household A Setup needs to be mapped as closely as possible onto the devices in Household B. As an example, imagine that Alice has created a Setup that allows her to see the image from the front door security camera on any of several display devices in her house. She will select the desired display device from the list when the Setup is run. Bob is Alice's brother, and he has just installed a security camera outside his back door, and would like to be able to view the output of his security camera on either the Living Room TV or the Bedroom TV, depending on his current location. As shown in Table 7-1 andTable 7-2, almost none of the devices in the two households are the same.

Source Component(s)	Destination Component(s)
ID: Webcam-ABCDEF123456 Name: Webcam Mfgr: Sony Model: DCX2010 Location: Front Porch Owner: Alice Obje Interfaces: Component, DataSource Source data types: video/mpeg, video/quicktime, application/x-obje-codebundle	ID: LCDTV-ABCDEF123456 Name: LCD TV Mfgr: Pioneer Model: TCZ420012 Location: Living Room Owner: Alice Obje Interfaces: Component, DataSink Sink data types: image/jpeg, image/png, video/avi, video/quicktime, application/x-obje-codebundle
	ID: TV-ABCDEF123456 Name: TV Mfgr: Sharp Model: Aquos 23.1156 Location: Office Owner: Alice Obje Interfaces: Component, DataSink Sink data types: image/jpeg, image/png, video/avi, video/quicktime, application/x-obje-codebundle
	ID: PictureFrame-ABCDEF123456 Name: Picture Frame Mfgr: CEIVA Model: 8.1 Location: Living Room Owner: Alice Obje Interfaces: Component, DataSink Sink data types: image/jpeg, image/png, application/x-obje-codebundle

Table 7-1: The devices in Alice's house that are used in a Setup that she wishes to share with Bob.

Source Component(s)	Destination Component(s)
ID: SecurityCamera-123456FEDCBA Name: Security Camera Mfgr: JVC Model: SC73BX01 Location: Back Door Owner: Administrator Obje Interfaces: Component, DataSource Source data types: application/flash-video, application/x-obje-codebundle	ID: PlasmaTV-123456FEDCBA Name: Plasma TV Mfgr: NEC Model: Vision 6134 Location: Family Room Owner: Administrator Obje Interfaces: Component, DataSink Sink data types: video/avi, video/quicktime, application/x-obje-codebundle
	ID: TV-123456FEDCBA Name: TV Mfgr: Sharp Model: Aquos 23.1156 Location: Master Bedroom Owner: Bob Obje Interfaces: Component, DataSink Sink data types: image/jpeg, image/png, video/avi, video/quicktime, application/x-obje-codebundle

Table 7-2: The devices in Bob's house that he would like to use in his version of Alice's Setup for viewing a security camera in multiple locations.

Because of the different devices used in each household, it is clear that a direct mapping of Alice's Setup will not work in Bob's house. If the Setup is only to be used within Alice's environment, the Setup would need only to keep track of the various device IDs in order to recreate the Setup every time Alice wishes to use it, as shown in Table 7-3. However, for it to be useful to Bob, Alice's Setup must be re-represented using an appropriate level of abstraction in order to be mapped onto Bob's network, as shown in Table 7-4 and Table 7-5. One possible solution is for Alice to perform an explicit "Export" operation to render the Setup sharable. This operation would rewrite the Setup to include as much information as possible about each of Alice's devices, such as the data types supported, the location, and the usage history, to increase the likelihood of matching each of the Setup's devices against devices in Bob's environment. With adequate information, it should be possible to find matches for each of the devices mentioned in Alice's Setup.

Setup: Show Security Camera on Nearby Display	
Source Component(s) *Selection rule: Use all*	Destination Component(s) *Selection rule: Show me and I will choose*
ID: Webcam-ABCDEF123456	ID: LCDTV-ABCDEF123456
	ID: TV-ABCDEF123456
	ID: PictureFrame-ABCDEF123456

Table 7-3: A simplified Setup to create the configuration shown in Table 7-1. This level of detail would be sufficient to support repeated use in Alice's house, but would not work in any other environment because the Setup only stores the IDs of Alice's devices.

Setup Name: Show Security Camera on Nearby Display	
Source Component(s) *Selection rule: Use all*	Destination Component(s) *Selection rule: Show me and I will choose*
~~ID: Webcam-ABCDEF123456~~ Name: Webcam Mfgr: Sony Model: DCX2010 Location: Front Porch ~~Owner: Alice~~ **Obje Interfaces: Component, DataSource** **Source data types: video/mpeg, video/quicktime, application/x-obje-codebundle**	~~ID: LCDTV-ABCDEF123456~~ Name: LCD TV Mfgr: Pioneer Model: TCZ420012 Location: Living Room ~~Owner: Alice~~ **Obje Interfaces: Component, DataSink** **Sink data types: image/jpeg, image/png, video/avi, video/quicktime, application/x-obje-codebundle**
	~~ID: TV-ABCDEF123456~~ Name: TV Mfgr: Sharp Model: Aquos 23.1156 Location: Office ~~Owner: Alice~~ **Obje Interfaces: Component, DataSink** **Sink data types: image/jpeg, image/png, video/avi, video/quicktime, application/x-obje-codebundle**
	~~ID: Webcam-ABCDEF123456~~ Name: Webcam Mfgr: Sony Model: DCX2010 Location: Front Porch ~~Owner: Alice~~ **Obje Interfaces: Component, DataSource** **Source data types: video/mpeg, video/quicktime, application/x-obje-codebundle**

Table 7-4: In order to share her Setup with Bob, the representation would need to be generalized to capture the characteristics of each device used and to eliminate the environment-specific information such as device IDs and owners.

Setup Name: Show Security Camera on Nearby Display	
Source Component(s) *Selection rule: Use all*	Destination Component(s) *Selection rule: Show me and I will choose*
~~ID: Webcam-ABCDEF123456~~ Name: Webcam Mfgr: Sony Model: DCX2010 Location: Front Porch ~~Owner: Alice~~ **Obje Interfaces: Component, DataSource** **Source data types: video/mpeg, video/quicktime, application/x-obje-codebundle**	~~ID: LCDTV-ABCDEF123456~~ ~~Name: LCD TV~~ ~~Mfgr: Pioneer~~ ~~Model: TCZ420012~~ ~~Location: Living Room~~ ~~Owner: Alice~~ **Obje Interfaces: Component, DataSink** **Sink data types: ~~image/jpeg, image/png, video/avi,~~ video/quicktime, application/x-obje-codebundle**

Table 7-5: An even more generalized version of Alice's Setup could be created by representing only the characteristics shared by *all* of the devices in both source and destination slots. When this Setup is invoked in Bob's environment, it will show all destination devices that support the Component and DataSink interfaces, and that support the video/quicktime and x-obje-codebundle Sink data types.

Even with rich information about Alice's devices, challenges still remain. What is needed for this scenario is not an exact match but the most relevant substitute. The appropriateness of a substitution candidate will be based on the degree of match among only a subset of each device's fields—for example the "ID" field should not be considered as relevant to the matching algorithm, whereas the "Source data types" and "Sink data types" fields are highly relevant. The locations of devices may be somewhat important in this particular case, but much less important than the data types supported. It is likely that the relative importance of each field will differ from one Setup to another. It may not be possible to identify the most relevant fields for each Setup automatically, and some user guidance may be required for Export, Import, or both. Investigating exactly how much of Setup sharing can be performed automatically, and how to best integrate user input into the process, will be necessary in order to enable Setup sharing across households.

7.4.1 Support for Wide-Area Sharing: Communities of Composition

Allowing users to share across homes is helpful in supporting friends and family, but enabling large-scale sharing across the wide area could prove even more valuable.

7.4.1.1 End-User Programming Communities of Practice

Bonnie Nardi's seminal studies of end-user programming practice demonstrated the importance of cooperative work in communities that employ end-user programming [109]. In her studies of office spreadsheet users and users of CAD software, she documents the emergence of local developers who are not (necessarily) professional programmers that provide expertise, assistance, and most importantly, template programs that can be adapted and modified by other end-users to suit their personal needs. This finding is echoed in work conducted by Wendy Mackay, also in the early 1990's who looked at the sharing and adaptation of mail sorting rules and UNIX configuration files [94].

The proposed work described in the previous section regarding Setups shared among friends and family is, in essence, aimed at supporting these types of communal practices. However, a more ambitious degree of sharing may be possible and has yet to be explored thoroughly in previous research. Might it be possible to support wide area communities of programming or compositional practice without the need for face-to-face collaboration? If such communities could be created and fostered, it could have enormous benefits to end-users, who may be able to benefit from the efforts of large numbers of peers who may have already generated a solution to whatever configuration problem they are facing. Large scale contributions to and use of a shared repository of Setups, filtered by the degree to which contributed Setups would match well with a user's own environments and ranked by means of Collaborative Filtering systems [123] have the potential to match users interests and their local network's capabilities with other users in similar situations who may have already created particularly useful Setups. At the same time, creating and supporting such large scale communities will present significant challenges, such as soliciting contributions from members, mapping such contributions to other users' networks, filtering out redundant contributions, and tracking and identifying community members' interests and usage patterns. Though these problems are substantial, the potential benefits are quite substantial as well, and I believe that further research should chart a course in this direction.

7.5 Summary

In previous chapters, I presented the results of my research into delivering an integrated user experience of ubiquitous computing. In this chapter, I discussed a set of areas that have not been addressed by my work, but that may prove to be important in further solidifying or extending the integrated user experience provided by the current versions of Obje and OSCAR. For these areas of future work, I have described why they are important and the initial steps one might take to address them. In the final chapter, I will return to the contributions of my work and discuss how they combine together to address the critical barriers to delivering an integrated user experience of ubicomp.

8 Conclusion

In the Introduction of this dissertation I described the emerging world of ubiquitous computing and discussed the challenges that this world poses for users. In particular I discussed the difficulty of presenting an integrated user experience of interacting with multiple devices and services in a highly networked and dynamic world. I described three problems with existing approaches to delivering end-user functionality in ubiquitous computing environments: piecemeal interoperability, piecemeal interaction, and sluggish adaptation. In this concluding chapter, I will review this dissertation's contributions in order to show how they address the three core user experience problems.

8.1 Thesis Statement

I will begin by returning to the thesis statement of this dissertation:

> *An integrated, yet flexible and customizable user experience of interacting with multiple heterogeneous devices and services is achievable through a combination of*
>
> - *a middleware framework that supports robust interoperability;*

- *mechanisms and end-user tools that allow ad hoc connections among distributed devices and services;*
- *framework- and application-level support for dynamically distributable control that allows users to monitor and control both individual devices and ongoing connections; and*
- • *end-user tools that support the discovery, connection, and control of individual devices as well as the creation, modification, and invocation of both temporary and reusable service compositions.*

The testbed system employed for the OSCAR user study, comprising as it did a number of Obje components running on a network with the Obje-enabled OSCAR client application to browse, select, connect, and control them, represents the achievement of an "integrated, yet flexible and customizable user experience." That the OSCAR user experience is *integrated* is evidenced by the fact that users were able to use a single, consistent, control client to accomplish a range of different tasks. That the experience is *flexible* is shown by the range of tasks represented, and the fact that users were able to create compositions to accomplish tasks using a user interface that would allow them to also create an even wider range of different compositions by simply re-applying the same techniques that were used in the study to a different set of services. The *customizability* of the experience derives from its flexibility. Each user can create a different set of compositions to encapsulate his or her preferred set of activities and means of carrying them out, and the end result will be a highly personal set of easy-to-invoke compositions that, in effect, represents a uniquely personal user interface to the full set of functionality available from his or her network of devices.

It should be clear by now that all of the constituent components of our sought-after user experience that are described in the thesis are represented in Obje and OSCAR, and it is precisely that combination of features that produce an integrated user experience. In the next section, I will recap the key contributions of the thesis and tie them back to both the thesis statement criteria and the original problems of piecemeal interoperability, piecemeal interaction, and sluggish adaptation that formed the launching point for this dissertation.

8.2 Contributions

1) *A service framework that supports robust interoperability and end-user composition.*

The Obje Framework provides *robust interoperability* by dictating a minimal set of *a priori* agreements among cooperating services and employing mobile code to allow one service to extend another's behavior at runtime. In addition to serving as a framework with interoperability guarantees that go beyond those offered by competing approaches, Obje defines simple, standard service interfaces that describe the roles that services can play in compositions, thus laying the groundwork for both *ad hoc connections* and *end-user composition*. Finally, Obje provides multiple mechanisms for services to deploy control user interfaces to clients, thus providing *support for dynamically distributable control*.

Obje plays a key role on addressing all three of the key user experience problems (*piecemeal interoperability*, *piecemeal interaction*, and *sluggish adaptation*), but without end-user applications to bring its benefits to users, it is powerless to affect the user experience. The Obje Display Mirror and OSCAR each provide a bridge by which Obje can extend its reach into the hands of the user.

2) *A case study of a shared display service that demonstrates how a persistent, networked, user-accessible service can provide advantages over the hardwired legacy system it replaced.*

While the Obje Display Mirror (ODM) experience provides benefits and contributions in its own right, with respect to the overarching goals of this dissertation, the primary contribution of the ODM is to provide an experience of *ad hoc connections* and *distributed control*. ODM users can connect to never before seen displays without any special configuration other than the one-time web-based client installation. Once the connection is established, the display service provides a control user interface that allows the user to not only control their own connection, but also receive the ability to monitor and control the connections of any simultaneous users. Through these features, the ODM is able to provide a partial solution to piecemeal interaction, primarily through providing users with an application client for controlling the destination and appearance of the information they are sharing within the same device (their laptop) as the information itself. However, as a specialized application that only allows users to accomplish a narrow range of tasks, it may actually contribute as much to the problem of *piecemeal interaction* as it does to its solution.

3) *A novel application and user interface for end-user composition in home media networks that can be used effectively by people with a range of technical skill to accomplish a variety of tasks.*

As an embodiment of an end-user composition system built atop a middleware framework that supports robust interoperability, ad hoc connections, dynamically distributable control, and support for both temporary and reusable compositions of devices and services, OSCAR provides a complete demonstration of this dissertation's thesis. It provides a working, testable example of the "integrated, yet flexible and customizable user experience" described in the thesis statement, and, indeed, a two-phase user study with 18 predominantly non-technical users showed that OSCAR was effective at achieving the overarching goal of an integrated, flexible, and customizable user experience that has been the aim of all of the research reported in this dissertation.

8.3 An Integrated User Experience of Ubiquitous Computing

The goal of this dissertation research has been to understand how to provide an integrated user experience of ubiquitous computing. I have reported my experiences designing and evaluating a framework, set of services, and end-user application to support such an integrated experience with a focus on the domain of home media. The experiences reported with respect to the Obje Display Mirror further suggest that such experiences can be delivered in the workplace environment, though a thorough exploration of integrated composition and control in such a domain has not yet been carried out. My work provides a solid foundation upon which further explorations of end-user service composition and seamless, integrated control of various types of heterogeneous environments can be carried out. In my own continuing research, I plan to address additional aspects of this problem, and I believe that there are substantial opportunities for additional perspectives to be applied and research to be conducted by a variety of designers and researchers. It is important that such research continues, as the difference between an enormous and ever-renewed bounty of new capabilities and a user experience of persistent frustration hangs in the balance.

Bibliography

1 Abowd, Gregory D. and Elizabeth D. Mynatt. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction.* **7**(1). pp. 29-58, 2000.

2 Abowd, Gregory D., Elizabeth D. Mynatt, and Tom Rodden. The Human Experience, *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, vol. 1(1): pp. 48-57, January, 2002.

3 Abrams, Marc, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. UIML: An Appliance-Independent XML User Interface Language. In Proceedings of *the 8th International World Wide Web Conference (WWW '99).* Toronto, Canada. pp. 1695-708, May 11-14, 1999.

4 Apple Computer, Inc., *Bonjour*, 2007. Apple Computer. http://www.apple.com/macosx/features/bonjour

5 Apple Computer, Inc., *iLife*, 2007. http://www.apple.com/ilife

6 Arnstein, Larry, Chia-Yang Hung, Robert Franza, Qing Hong Zhou, Gaetano Borriello, Sunny Consolvo, and Jing Su. Labscape: A Smart Environment for the Cell Biology Laboratory. *IEEE Pervasive Computing.* **1**(3). pp. 13-21, 2002.

7 Balfanz, Dirk, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking To Strangers: Authentication in Ad-Hoc Wireless Networks. In Proceedings of *Network and Distributed System Security Symposium (NDSS '02).* San Diego, CA, USA, February, 2002.

8 Beyer, Hugh and Karen Holtzblatt, *Contextual Design: Defining Customer-Centered Systems.* San Francisco, CA, USA: Morgan Kaufmann. 1998.

9 Blackwell, Alan F. and Rob Hague. AutoHAN: An Architecture for Programming the Home. In Proceedings of *the IEEE 2001 Symposium on Human-Centric Languages and Environments (HCC '01).* Stresa, Italy. pp. 150-57, Sept. 5-7, 2001.

10 Block, Ryan, *Zune review*, 2006. http://www.engadget.com/2006/11/15/zune-review/

11 Bluetooth Consortium, *Specification of the Bluetooth System, Version 1.1 Core*, February 22 2001. http://www.bluetooth.com

12 Bolin, Michael, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and Customization of Rendered Web Pages. In Proceedings of *the 18th Annual ACM Symposium on User Interface Software and Technology (UIST 2005)*. Seattle, WA. pp. 163-72, 2005.

13 Borenstein, N. and N. Freed, *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Messages*. Internet RFC 1341, June 1992.

14 Bowker, Geof, Information Mythology and Infrastructure, in *Information Acumen: The Understanding and Use of Knowledge in Modern Business*, L. Bud-Frierman, Editor. Routledge: London. pp. 231-47, 1994.

15 Broll, Gregor, Enrico Rukzio, and Björn Wedi, Authoring support for mobile interaction with the real world, in *the 5th Annual Conference on Pervasive Computing (Pervasive 2007)*. 2007: Toronto, Canada.

16 Brooke, John, SUS: A quick and dirty usability scale, in *Usability Evaluation in Industry*, P.W. Jordan, B. Thomas, B.A. Weerdmeester, and I.L. McClelland, Editors. Taylor and Francis: London. pp. 189-94, 1996.

17 Campbell, Andrew, Geoff Coulson, and David Hutchison. A Quality of Service Architecture. *ACM SIGCOMM Computer Communication Review.* **24**(2). pp. 6-27, 1994.

18 Carnoy, David, *CNET editors' review: Logitech Harmony 880*, 2005. CNET. http://reviews.cnet.com/Logitech_Harmony_880/4505-7900_7-31337419.html

19 Carriero, Nicholas and David Gelernter. Linda in Context. *Communications of the ACM.* **32**(4). pp. 444-58, 1989.

20 CEIVA Logic Inc., *CEIVA Official Site, Learn More About the CEIVA Digital Photo Frame*, 2007. http://www.ceiva.com/lmore/dpr/dpr.jsp

21 Cheshire, Stuart and Marc Krochmal, *DNS-Based Service Discovery*, 2006. IETF. http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt

22 Cheshire, Stuart and Marc Krochmal, *Multicast DNS*, IETF 2004. http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt

23 Christensen, Erik, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana, *Web Services Description Language (WSDL) 1.1*, 2001. W3C. http://www.w3.org/TR/wsdl

24 Colin, David, *NEC's Wireless MT1065*, 2003. ProjectorCentral.com. http://www.projectorcentral.com/wireless_nec_mt1065.htm

25 Consolvo, Sunny, Larry Arnstein, and B. Robert Franza. User Study Techniques in the Design and Evaluation of a Ubicomp Environment. In Proceedings of *the 4th International Conference on Ubiquitous Computing*. Goteborg, Sweden. pp. 73-90, September 29 - October 1, 2002.

26 Consolvo, Sunny and Miriam Walker. Using the Experience Sampling Method to Evaluate Ubicomp Applications, *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, vol. 2(2): pp. 24-31, April - June, 2003.

27 Cooper, Alan, *The Inmates are Running the Asylum*. Indiannapolis, IN, USA: Macmillan Publishing Co. 1999.

28 Csikszentmihalyi, Mihaly and R. Larson. Validity and Reliability of the Experience-Sampling Method. *Journal of nervous and mental disease.* **175**(9). pp. 526-36, 1987.

29 Cypher, Allen. Eager: Programming Repetitive Tasks by Example. In Proceedings of *the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '91).* New Orleans, LA. pp. 33-39, April 28-May 2, 1991.

30 Czerwinski, Steven E., Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. An Architecture for a Secure Discovery Service. In Proceedings of *the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99).* Seattle, WA. pp. 24-35, August 15-19, 1999.

31 Davis, Richard C., James A. Landay, Victor Chen, Jonathan Huang, Rebecca B. Lee, Francis C. Li, James Lin, Charles B. III Morrey, Ben Schleimer, Morgan N. Price, and Bill N. Schilit. NotePals: Lightweight Note Sharing by the Group, for the Group. In Proceedings of *the ACM Conference on Human Factors in Computing Systems (CHI '99).* Pittsburgh, PA. pp. 338-45, May 15-20, 1999.

32 Dey, Anind K., Daniel Salber, and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting Rapid Prototyping of Context-Aware Applications. *Human Computer Interaction Journal.* **16**(2-4). pp. 97-166, 2001.

33 Dey, Anind K., Timothy Sohn, Sara Streng, and Justin Kodama. iCAP: Interactive Prototyping of Context-Aware Applications. In Proceedings of *the 4th International Conference on Pervasive Computing (Pervasive '06)*. Dublin, Ireland, May, 2006.

34 Digital Living Network Alliance, *DLNA Technology*, 2007. http://www.dlna.org/en/consumer/learn/technology/

35 Digital Living Network Alliance, *Overview and Vision White Paper*, 2004. http://www.dlna.org/about/DLNA_Overview.pdf

36 Dourish, Paul, W. Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Michael Salisbury, Douglas B. Terry, and James Thornton. Extending Document Management Systems with Active Properties. *ACM Transactions on Information Systems*, 2000.

37 Ducheneaut, Nicolas, Trevor F Smith, James "Bo" Begole, Mark W. Newman, and Chris Beckman. The Orbital Browser: Composing Services Using only Rotation and Selection. In Proceedings of *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI 2006)*. Montreal, Quebec. pp. 321-26, April 22-27, 2006.

38 Edwards, W. Keith. Discovery Systems in Ubiquitous Computing. *IEEE Pervasive Computing.* **5**(2). pp. 70-77, 2006.

39 Edwards, W. Keith, Victoria Bellotti, Anind K. Dey, and Mark W. Newman. Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Infrastructure. In Proceedings of *the 2003 Conference on Human Factors in Computing Systems (CHI 2003)*. Fort Lauderdale, FL USA. pp. 297-304, April 5-10, 2003.

40 Edwards, W. Keith, Mark Newman, Jana Z Sedivy, and Trevor F Smith. Supporting Serendipitous Integration in Mobile Computing Environments. *International Journal of Human-Computer Studies.* **60**(5-6). pp. pp. 666-700, 2004.

41 Edwards, W. Keith, Mark W. Newman, Jana Z. Sedivy, Trevor F Smith, Dirk Balfanz, D. K. Smetters, H. Chi Wong, and Shahram Izadi. Using Speakeasy for Ad Hoc Peer-

to-Peer Collaboration. In Proceedings of *the ACM Conference on Computer Supported Cooperative Work (CSCW '02).* New Orleans, LA USA, November 16-20, 2002.

42 Edwards, W. Keith, Mark W. Newman, Jana Z. Sedivy, Trevor F Smith, and Shahram Izadi. Challenge: Recombinant Computing and the Speakeasy Approach. In Proceedings of *the 8th ACM International Conference on Mobile Computing and Networking (MobiCom 2002).* Atlanta, GA USA. pp. 279-86, September 23-28, 2002.

43 Edwards, W. Keith, Mark W. Newman, Trevor F Smith, Jana Z. Sedivy, and Shahram Izadi. An Extensible Set-top Box Platform for Home Media Applications. *IEEE Transactions on Consumer Electronics.* **51**(4). pp. 1175-81, 2005.

44 eHomeUpgrade, *Archive: Streaming Media Devices*, 2007. http://www.ehomeupgrade.com/archive/streaming_media_devices

45 Escobar, Julio, Debra Deutsch, and Craig Partridge, *A Multi-Service Flow Synchronization Protocol*, BBN Systems and Technologies Division, March 1991.

46 Falcone, John P., *Editors' top network media players*, 2006. CNET. http://reviews.cnet.com/4323-6531_7-6509113.html

47 Fox, Armando, Brad Johanson, Pat Hanrahan, and Terry Winograd. Integrating Information Appliances into an Interactive Workspace. *IEEE Computer Graphics & Applications.* **20**(3). pp. 54-65, 2000.

48 Froehlich, Jon, Mike Y. Chen, Sunny Consolvo, Beverly Harrison, and James A. Landay. MyExperience: a System for *in situ* Tracing and Capturing of User Feedback on Mobile Phones. In Proceedings of *the 5th International Conference on Mobile Systems, Applications and Services (Mobisys '07).* San Juan, Puerto Rico. pp. 57-70, 2007.

49 Garfinkel, Harold, *Studies in Ethnomethodology*. Englewood Cliffs, NJ: Prentice- Hall. 1967.

50 Goland, Y. Y., T. Cai, P. Leach, Y. Gu, and S. Albright, *Simple Service Discovery Protocol/1.0: Operating Without an Arbiter*, Internet Engineering Task Force Internet Draft 1999. http://www.upnp.org/draft_cai_ssdp_v1_03.txt

51 Gottman, John Mordechai and Anup Kumar Roy, *Sequential Analysis: A Guide for Behavioral Researchers*: Cambridge University Press. 1990.

52 Gribble, Steven D., Matt Welsh, J. Robert von Behren, Eric A. Brewer, David Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, Anthony Joseph, Randy H. Katz, Z. M. Mao, S. Ross, and B. Zhao. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Computer Networks.* **35**(4). pp. 473-97, 2001.

53 Grimes, Richard, *Professional DCOM Programming*: Wrox Press. 1997.

54 Grimm, Robert, Janet Davis, Eric Lemar, Adam MacBeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall. System Support for Pervasive Applications. *ACM Transactions on Computer Systems.* **22**(4). pp. 421-86, 2004.

55 Grinter, Rebecca E., W. Keith Edwards, Mark W. Newman, and Nicolas Ducheneaut. The Work to Make a Home Network Work. In Proceedings of *The European Conference on Computer-Supported Cooperative Work (ECSCW '05).* Paris, France, 2005.

56 Grudin, Jonathan. Why Groupware Applications Fail: Problems in Design and Evaluation. *Office: Technology and People.* **4**(3). pp. 245-64, 1989.

57 Grudin, Jonathan and John Pruitt. Personas, Participatory Design and Product Development: An Infrastructure for Engagement. In Proceedings of *the Participatory Design Conference (PDC 2002).* Malmo, Sweeden, June, 2002.

58 Gudgin, Martin, Marc Hadley, Noah Mendelsohn, Jean-Jaques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon, *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, 2007. http://www.w3.org/TR/soap12-part1/

59 Hague, Rob and Peter Robinson. Multi-lingual End-user Programming with XML. In Proceedings of *the 3rd Program Visualization Workshop (PVW '04).* Warwick, UK. pp. 34-40, 2004.

60 Hall, Edward T., *The Hidden Dimension.* New York: Anchor Books. 1966.

61 Harmonia Inc., *User Interface Modelling Language 2.0 Draft Specification*, 2000. http://www.uiml.org/specs/uiml2/index.htm

62 Hartmann, Björn, Leslie Wu, Kevin Collins, and Scott R. Klemmer. Programming by a Sample: Rapidly Prototyping Web Applications with d.mix. In Proceedings of *the 20th*

Annual ACM Symposium on User Interface Software and Technology (UIST 2007). Newport, RI USA. pp. 241-50, 2007.

63 Hodes, Todd and Randy H. Katz. A Document-based Framework for Internet Application Control. In Proceedings of *the 2nd USENIX Symposium on Internet Technologies and Systems (USITS '99)*. Boulder, CO USA. pp. 59-70, October, 1999.

64 Hof, Robert D. Mix, Match, and Mutate, *BusinessWeek*, July 25, 2005.

65 Hong, Jason I., *An Architecture for Privacy-Sensitive Ubiquitous Computing*, Unpublished Dissertation, University of California, Berkeley, Electrical Engineering and Computer Sciences Department, Berkeley, CA, 2005.

66 Hong, Jason I. and James A. Landay. An Architecture for Privacy-sensitive Ubiquitous Computing. In Proceedings of *The 2nd International Conference on Mobile Systems, Applications, and Services (Mobisys 2004)*. Boston, MA, USA. pp. 177-89, June 6-9, 2004.

67 Huang, Andrew C., Benjamin C. Ling, John Barton, and Armando Fox. Making Computers Disappear: Appliance Data Services. In Proceedings of *the 7th ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom 2001)*. Rome, Italy. pp. 108-21, July, 2001.

68 Hudson, Scott E., James Fogarty, Christopher G. Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny C. Lee, and Jie Yang. Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study. In Proceedings of *the 2003 Conference on Human Factors in Computing Systems (CHI 2003)*. Fort Lauderdale, FL USA. pp. 257-64, April 5-10, 2003.

69 Hull, Richard, Ben Clayton, and Tom Melamed. Rapid Authoring of Mediascapes. In Proceedings of *the 6th Annual Conference on Ubiquitous Computing (Ubicomp 2004)*. Nottingham, UK. pp. 125-42, 2004.

70 Humble, Jan, Andy Crabtree, Terry Hemmings, Karl-Petter Åkesson, Boriana Koleva, Tom Rodden, and Pär Hansson. "Playing with the Bits": User-configuration of Ubiquitous Domestic Environments. In Proceedings of *UbiComp: The Fifth International Conference on Ubiquitous Computing (Ubicomp 2003)*. Seattle, Washington. pp. 256-63, October 12-15, 2003.

71 IBM Services Architecture Team, *Web Services Architecture Overview: the Next Stage of Evolution for e-business*, 2000. IBM. http://www.ibm.com/developerworks/library/w-ovr/

72 Internet Engineering Task Force (IETF) Zeroconf Working Group, *Zero Configuration Networking (Zeroconf)*, 2005. http://www.zeroconf.org

73 Intille, Stephen S., John Rondoni, Charles Kukla, Isabel Anacona, and Ling Bao. A Context-aware Experience Sampling Tool. In Proceedings of *Extended Abstracts of the Conference on Human Factors and Computing Systems (CHI 2003)*. Fort Lauderdale, FL USA. pp. 972-73, April 5-10, 2003.

74 Jeronimo, Michael and Jack Weast, *UPnP Design by Example*: Intel Press. 2003.

75 Johanson, Brad, Armando Fox, and Terry Winograd. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, vol. 1(2), April - June, 2002.

76 Johanson, Brad, Emre Kiciman, and Armando Fox, Moving Data and Interfaces in an Interactive Workspace, in *the Workshop on Infrastructure for Smart Devicdes at Handheld and Ubiqitous Computing (HUC 2000)*. 2000: Bristol, UK.

77 Joshi, James, Elisa Bertino, and Arif Ghafoor. Temporal Hierarchies and Interitance Semantics for GTRBAC. In Proceedings of *the 7th ACM Symposium on Access Control Models and Technologies (SACMAT '02)*. Monterey, CA. pp. 74-83, June 3-4, 2002.

78 Kay, Alan C. The Early History of Smalltalk. In Proceedings of *the 2nd SIGPLAN Conference on the History of Programming Languages* Cambridge, MA. pp. 69-95, April 20-23, 1993.

79 Kelleher, Caitlin and Randy Pausch. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys.* **37**(2). pp. 83-137, 2005.

80 Kiciman, Emre and Armando Fox. Using Dynamic Mediation to Integrate COTS Entities in a Ubiquitous Computing Environment. In Proceedings of *the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC 2000)*. Bristol, UK. pp. 211-26, Sept. 25-27, 2000.

81 Kiciman, Emre, Laurence Melloul, and Armando Fox, Position Summary: Towards Zero-code Service Composition, in *the 8th Workshop in Hot Topics in Operating Systems (HotOS VIII)*. 2001: Elmau/Oberbayern, Germany.

82 Kiczales, Gregor, Jim des Rivieres, and Daniel G. Bobrow, *The Art of the Metaobject Protocol*. Cambridge, Massachusetts: MIT Press. 1991.

83 Kindberg, Tim and John Barton. A Web-based Nomadic Computing System. *Computer Networks*. **35**. pp. 443-56, 2001.

84 Kindberg, Tim and Armando Fox. System Software for Ubiquitous Computing. *IEEE Pervasive Computing*. **1**(1). pp. 70-81, 2002.

85 LaMonica, Martin, *Teqlo: Not a programmer? Not a problem*, 2006. cnet News.com. http://news.com.com/8301-10784_3-6125190-7.html

86 Lau, Tessa and Daniel S. Weld. Programming by Demonstration: An Inductive Learning Formulation. In Proceedings of *the International Conference on Intelligent User Interfaces (IUI '99)*. Redondo Beach, CA. pp. 145-52, 1999.

87 Lewis, Clayton and John Reiman, *Task-Centered User Interface Design: A Practical Introduction*, 1993. Shareware Book. http://hcibib.org/tcuid

88 Li, Yang, Jason I. Hong, and James A. Landay. Topiary: a Tool for Prototyping Location-enhanced Applications. In Proceedings of *the 17th Annual ACM Symposium on User Interface Software and Technology (UIST 2004)*. Santa Fe, NM USA. pp. 217-26, October 24-27, 2004.

89 Lieberman, Henry and José Espinosa. A Goal-oriented Interface to Consumer Electronics Using Planning and Commonsense Reasoning In Proceedings of *the ACM Conference on Intelligent User Interfaces (IUI 2006)*. Sydney, Australia. pp. 226-33, Jan. 20-Feb. 1, 2006.

90 Logo Computer Systems, Inc., *The Logo Programming Language*, 1995. available at http://microworlds.com

91 Lohse, Marco and Philipp Slusallek. Towards Automatic Setup of Distributed Multimedia Applications. In Proceedings of *International Conference on Internet and Multmedia Systems and Applications*. Innsbruck, Austria. pp. 359-64, 2005.

92 Lynch, Michael, E. Livingstone, and E. Garfield, Temporal Order in Laboratory Work, in *Science Observed: Perspectives on the Social Study of Science*, K.D. Knoll-Centina and M. Mulkay, Editors. Sage: London, 1983.

93 Mac News Network, *WiJET.Video: 802.11g wireless display adapter*, 2004. http://www.macnn.com/articles/04/01/08/wijet.video.802.11g/

94 Mackay, Wendy E. Patterns of Sharing Customizable Software. In Proceedings of *the 1990 ACM Conference on Computer-supported Cooperative Work (CSCW '90)*. Los Angeles, CA USA. pp. 209-21, 1990.

95 Mainwaring, Scott, Michelle F. Chang, and Ken Anderson. Infrastructures and their Discontents: Implications for Ubicomp. In Proceedings of *the 6th International Conference on Ubiquitous Computing*. Nottingham, UK. pp. 418-32, September 7-10, 2004.

96 Mansley, Kieran, Alastair Beresford, and David Scott. The Carrot Approach: Encouraging Use of Location Systems. In Proceedings of *Ubicomp 2004*. Nottingham, UK. pp. 366-83, 2004.

97 Mao, Zhuoqing Morley and Randy H. Katz. Achieving Service Portability Using Self-Adaptive Data Paths. *IEEE Communications.* **40**(1). pp. 108-14, 2002.

98 Masuoka, Ryusuke, Bijan Parsia, and Yannis Labrou. Task Computing - the Semantic Web Meets Pervasive Computing. In Proceedings of *2nd International Semantic Web Conference (ISWC2003)*. Sanibel Island, Florida, USA, 20-23 October, 2003.

99 Mernik, Marjan, Jan Heering, and Anthony M. Sloane. When and How to Develop Domain-specific Languages. *ACM Computing Surveys.* **37**(4). pp. 316-44, 2005.

100 Messer, Alan, Henry Song, Praveen Kumar, Phuong Nguyen, Anugeetha Kunjithapatham, and Mithun Sheshagiri. InterPlay: A Middleware for Integration of Devices, Services and Contents in the Home Networking Environment. In Proceedings of *3rd IEEE Consumer Communications and Networking Conference (CCNC 2006)*. Las Vegas, NV, USA. pp. 1083-87, 8-10 Jan, 2006.

101 Microsoft Corp., *Express - Visual Basic - Easy to Use*, 2007. http://msdn2.microsoft.com/en-us/express/aa718406.aspx

102 Microsoft Corp., *Microsoft Popfly*, 2007. http://www.popfly.com/Overview/

103 Microsoft Corp., *What Is .NET?*, 2005. http://www.microsoft.com/net/basics.mspx

104 Microsoft Corp., *Windows Media Player*, 2007. http://www.microsoft.com/windows/windowsmedia

105 Microsoft Corp., *Working Remotely with Windows XP*, 2005. http://www.microsoft.com/windowsxp/using/mobility/default.mspx

106 Monson-Haefel, Richard, *Enterprise Java Beans*. 3 ed: O'Reilly. 2001.

107 Mynatt, Elizabeth D., *everyday computing lab*, 2004. Georgia Institute of Technology. http://www.cc.gatech.edu/fce/ecl/

108 Nakao, Akihiro, Larry Peterson, and Andy Bavier. Constructing end-to-end paths for playing media objects. *Computer Networks*. **38**(2002). pp. 373-89, 2002.

109 Nardi, Bonnie, *A Small Matter of Programming: Perspectives on End User Computing*. Cambridge, MA: MIT Press. 1993.

110 Newman, Mark W., Nicolas Ducheneaut, W. Keith Edwards, Jana Z. Sedivy, and Trevor F Smith. Supporting the unremarkable: experiences with the Obje Display Mirror. *Personal and Ubiquitous Computing*. **Published Online**(http://www.springerlink.com/content/t0p5l7l3p5r400gw/), 2006.

111 Newman, Mark W., Shahram Izadi, W. Keith Edwards, Jana Z. Sedivy, and Trevor F. Smith. User interfaces when and where they are needed: an infrastructure for recombinant computing. In Proceedings of *the 15th annual ACM symposium on User interface software and technology*. pp. 171-80, 2002.

112 Newman, Mark W., Jana Z. Sedivy, Christine M. Neuwirth, W. Keith Edwards, Jason I. Hong, Shahram Izadi, Karen Marcelo, and Trevor F Smith. Designing for Serendipity: Supporting End-user Configuration of Ubiquitous Computing Environments. In Proceedings of *the International Conference on Designing interactive Systems (DIS 2002)*. London, UK. pp. 147-56, 2002.

113 Nichols, Jeffrey, Duen Horng Chau, and Brad A. Myers. Demonstrating the Viability of Automatically Generated User Interfaces. In Proceedings of *the ACM Conference on Human Factors in Computing Systems (CHI '07)*. San Jose, CA. pp. 1283-92, 2007.

114 Nichols, Jeffrey, Brad A. Myers, MIchael Higgins, Thomas K. Hughes, Roni Rosenfeld, and Mathilde Pignol. Generating Remote Control Interfaces for Complex Appliances. *CHI Letters: Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology (UIST 2002).* **4**(2). pp. 161-70, 2002.

115 Nichols, Jeffrey, Brad A. Myers, and Brandon Rothrock. UNIFORM: Automatically Generating Consistent Remote Control User Interfaces. In Proceedings of *the 24th Annual ACM Conference on Human Factors in Computing Systems (CHI 2006).* Montreal, Canada. pp. 611-20, April 22-26, 2006.

116 Nichols, Jeffrey, Brandon Rothrock, Duen Horng Chau, and Brad A. Myers. Huddle: Automatically Generating Interfaces for Systems of Multiple Connecting Appliances. In Proceedings of *the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06).* Montreux, Switzerland, October 15-18, 2006.

117 Nullsoft, *SHOUTcast - Documentation*, 2007. http://www.shoutcast.com/support/docs/

118 Nunamaker, Jay F., A. R. Dennis, and J. S. Valencich. Electronic Meeting Systems to Support Group Work. *Communications of the ACM.* **34**(7). pp. 40-61, 1991.

119 Object Management Group, *CORBA: The Common Object Request Broker Architecture, Rev. 2.0*, July, 1995 1995.

120 Ockerbloom, John, *Mediating Among Diverse Data Formats*, Carnegie Mellon University Technical Report CMU-CS-98-102, Computer Science, Pittsburgh, PA, 1998.

121 Ponnekanti, Shankar R., Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd. ICrafter: A Service Framework for Ubiquitous Computing Environments. In Proceedings of *the 3rd International Conference on Ubiquitous Computing (Ubicomp 2001).* Atlanta, GA USA. pp. 56-75, September, 2001.

122 Rangan, P. Venkat, Srinivas Ramanathan, and Thomas Kaeppner. Performance of Inter-Media Sychronization in Distributed and Heterogeneous Multimedia Systems. *Computer Networks and ISDN Systems.* **27**(4). pp. 549-65, 1995.

123 Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: an Open Architecture for Collaborative Filtering of netnews. In Proceedings

of *the ACM Conference on Computer-supported Cooperative Work (CSCW '94)*. Chapel Hill, NC. pp. 175-86, October 22-26, 1994.

124 Rettig, Marc. Prototyping for Tiny Fingers, *Communications of the ACM*, 1994.

125 Richardson, T., Q. Stafford-Fraser, K. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*. **2**(1), 1998.

126 Ritchie, John and Thomas Kuchnel, *UPnP AV Architecture 0.83*, 2002. UPnP Forum. http://www.upnp.org/standardizeddcps/documents/UPnPAvArchtiecture0.83.pdf

127 Rodden, Tom, Andy Crabtree, Terry Hemmings, Boriana Koleva, Jan Humble, Karl-Petter Akesson, and Par Hansson. Between the dazzle of a new building and its eventual corpse: assembling the ubiquitous home. In Proceedings of *the International Conference on Designing Interactive Systems (DIS 2004)*. Cambridge, MA, USA. pp. 71-80, 2004.

128 Rode, Jennifer A., Eleanor F. Toye, and Alan F. Blackwell. The Domestic Economy: a Broader Unit of Analysis for End User Programming. In Proceedings of *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI 2005)*. Portland, OR USA. pp. 1757-60, 2005.

129 Rode, Jennifer A., Eleanor F. Toye, and Alan F. Blackwell. The Fuzzy Felt Ethnography--Understanding the Programming Patterns of Domestic Appliances. *Personal and Ubiquitous Computing*. **8**. pp. 161-76, 2004.

130 Rogers, Everett, *Diffusion of Innovations*. 1962.

131 Román, Manuel, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, and Klara Nahrstedt. Gaia: a Middleware Infrastructure to Enable Active Spaces, *IEEE Pervasive Computing*, vol. 1(4): pp. 74-83, Oct.-Dec., 2002.

132 Rose, M., Internet Engineering Task Force, RFC 3080: The Blocks Extensible Exchange Protocol Core, I.E.T.F. (IETF), Editor. 2001.

133 Royal Philips Electronics, *Pronto* 2007. http://www.pronto.philips.com/

134 Salber, Daniel, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-enabled Applications. In Proceedings of *the ACM Conference on Human Factors in Computing Systems (CHI 1999)*. Pittsburgh, PA USA. pp. 434-41, May 15-20, 1999.

135 Scheifler, Robert W. and Jim Gettys. X Window System. *ACM Transactions on Graphics.* **5**(2). pp. 79-109, 1986.

136 Smetters, D. K., Dirk Balfanz, Glenn Durfee, Trevor F Smith, and Kyung-Hee Lee. Instant Matchmaking: Simple Secure Virtual Extensions to Ubiquitous Computing Environments. In Proceedings of *the 8th International Conference on Ubiquitous Computing (Ubicomp 2006).* Orange County, CA. pp. 477-94, September 17-21, 2006.

137 Snyder, Carolyn, *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*: Morgan Kaufmann. 2003.

138 Sousa, João Pedro and David Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In Proceedings of *the 3rd Working IEEE/IFIP Conference on Software Architecture*. pp. 29-43, August 25-31, 2002.

139 Star, S.L. The Ethnography of Infrastructure. *American Behavioral Scientist.* **43**(3). pp. 377-91, 1999.

140 Streitz, Norbert A., Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-LAND: an Interactive Landscape for Creativity and Innovation. In Proceedings of *the SIGCHI Conference on Human factors in computing systems (CHI '99).* Pittsburgh, Pennslyvania, USA. pp. 120-27, 1999.

141 Suchman, Lucy A., *Plans and Situated Actions: The Problem of Human-Computer Communication*. New York: Cambridge University Press. 1987.

142 Sun Microsystems, *Java EE at a Glance*, 2007. Sun Microsystems. http://java.sun.com/javaee/

143 Sun Microsystems, *Jini Discovery and Join Specification*, January 1999.

144 Sun Microsystems, *The Java Programming Language*, 1997. http://java.sun.com/javase/

145 TabletKiosk, *Sahara Slate PC i215 Touch-iT Tablet PC*, 2006. http://www.tabletkiosk.com/tkstore/pc/viewPrd.asp?idcategory=17&idproduct=68

146 TechSmith, *Camtasia Studio*, 2007. http://www.techsmith.com/camtasia.asp

147 TiVo, Inc., *TiVo*, 2007. http://www.tivo.com

148 Tolmie, Peter, James Pycock, Tim Diggins, Allan MacLean, and Alain Karsteny. Unremarkable Computing. In Proceedings of *the ACM Conference on Human Factors in Computing Systems (CHI '02).* Minneapolis, MN USA. pp. 399-406, 2002.

149 Truong, Khai N. and Gregory D. Abowd. INCA: Architectural Support for Building Automated Capture & Access Applications. In Proceedings of *The 2nd International Conference on Pervasive Computing (Pervasive 2004).* Vienna, Austria. pp. 140-57, April 21-23, 2004.

150 Truong, Khai N., Elaine M. Huang, and Gregory D. Abowd. CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In Proceedings of *the 6th International Conference on Ubiquitous Computing (Ubicomp 2004).* Nottingham, UK. pp. 143-60, 2004.

151 Turoff, M. Computer-Mediated Communication Requirements for Group Support. *Journal of Organizational Computing.* **1**(1). pp. 85-113, 1991.

152 Universal Description Discovery and Integration Consortium, *UDDI Technical Whitepaper*, September 6 2000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

153 UPnP Forum, *UPnP Device Architecture*, UPnP Forum, June 2000. http://www.upnp.org/download/UPnPDA10_20000613.htm

154 Van Kleek, Max, Kai Kunze, Kurt Partridge, and James "Bo" Begole. OPF: A Distributed Context-Sensing Framework for Ubiquitous Computing Environments. In Proceedings of *the 3rd International Symposium on Ubiqutous Computing Systems (UCS 2006).* Seoul, Korea. pp. 82-97, October 11-13, 2006.

155 Vanderheiden, Gregg C. and Gottfried Zimmermann. Use of User Interface Sockets to Create Naturally Evolving Intelligent Environments. In Proceedings of *the 11th International Conference on Human-Computer Interaction (HCI International 2005).* Las Vegas, NV, USA, July 22-27, 2005.

156 Venners, Bill, *The ServiceUI API Specification, Version 1.1beta3*, 2002. http://www.artima.com/jini/serviceui/Spec.html

157 Voida, Stephen, W. Keith Edwards, Mark W. Newman, Rebecca E. Grinter, and Nicolas Ducheneaut. Share and Share Alike: Exploring the User Interface Affordances of File Sharing. In Proceedings of *the ACM Conference on Human Factors in Computing Systems (CHI 2006)*. Montreal, Quebec. pp. 221-30, 2006.

158 Waldo, Jim. The Jini Architecture for Network-centric Computing, *Communications of the ACM*, vol. 42(7): pp. 76-82, July, 1999.

159 Walkenbach, John, *Excel 2007 Formulas*. Indiannapolis, IN: Wiley Publishing, Inc. 2007.

160 Weiser, Mark. The Computer for the Twenty-First Century, *Scientific American* pp. 94-100, 1991.

161 Weiser, Mark and John Seely Brown. Designing Calm Technology. *PowerGrid Journal.* **1.01**(1), 1996.

162 White, James E. A High-level Framework for Network-based Resource Sharing. In Proceedings of *the National Computer Conference*. pp. 561-70, June 6-7, 1975.

163 Wisneski, C., Hiroshi Ishii, A. Dahley, M. Gorbett, S. Brave, B. Ullmer, and P. Yarin. Ambient Displays: Turning Architectural Space into an Interface between People and Digital Information. In Proceedings of *the 1st International Workshop on Cooperative Buildings (CoBuild '98)*. Darmstadt, Germany. pp. 22-32, 1998.

164 Wollrath, A., R. Riggs, and J. Waldo. A Distributed Object Model for the Java System. *USENIX Computing Systems.* **9**, 1996.

165 Wolpin, Steward, *Sony RM-AV3000 Integrated Remote Commander*, 2003. http://reviews.cnet.com/remote-controls/sony-rm-av3000-integrated/4505-7900_7-20605050.html

166 Wong, Jeffrey and Jason I. Hong. Making Mashups with Marmite: Re-purposing Web Content through End-User Programming. In Proceedings of *the ACM Conference on Human Factors in Computing Systems (CHI 2007)*. pp. 1435-44, 2007.

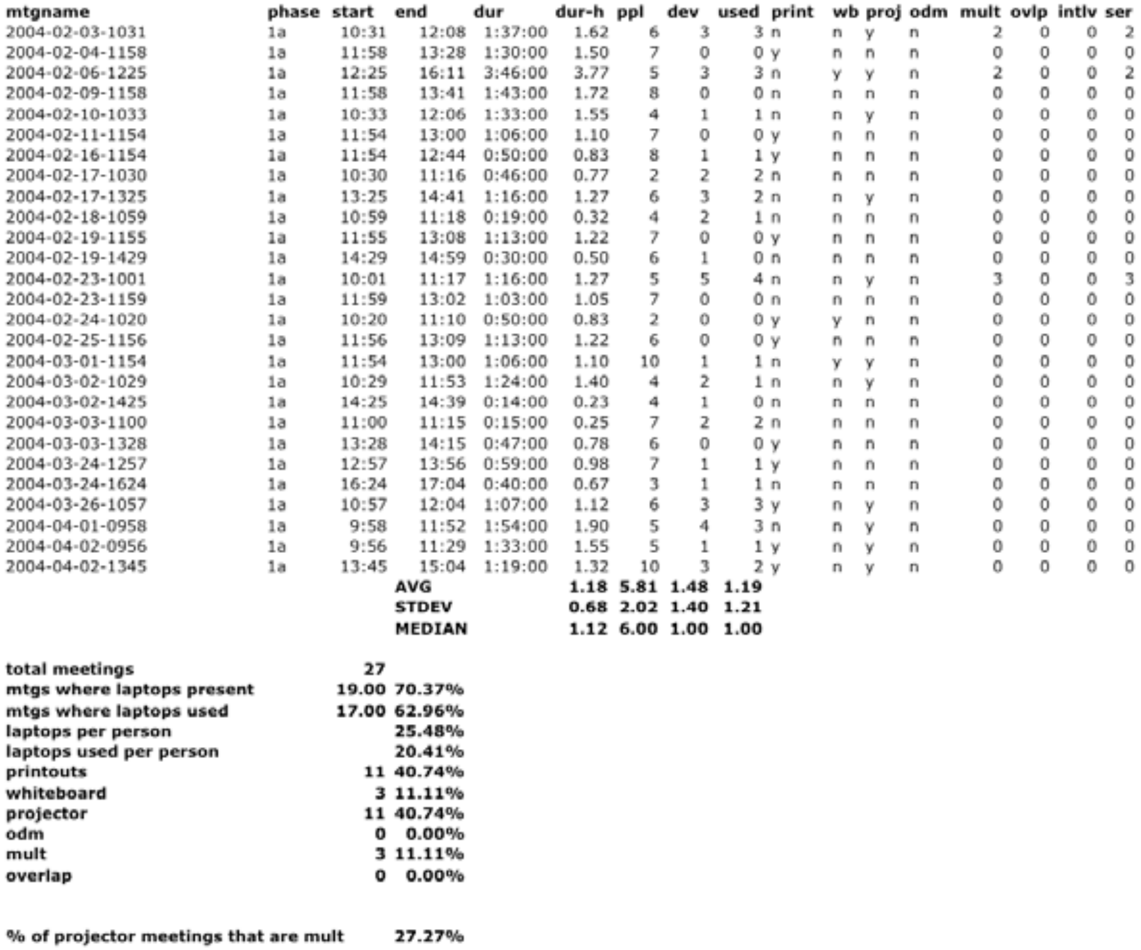
167 Yahoo! Inc., *Welcome to Flickr*, 2007. http://flickr.com

168 Yahoo! Inc., *Yahoo! Pipes*, 2007. http://pipes.yahoo.com/

169 YouTube, LLC, *YouTube - Broadcast Yourself*, 2007. http://youtube.com

A Obje Display Mirror Data

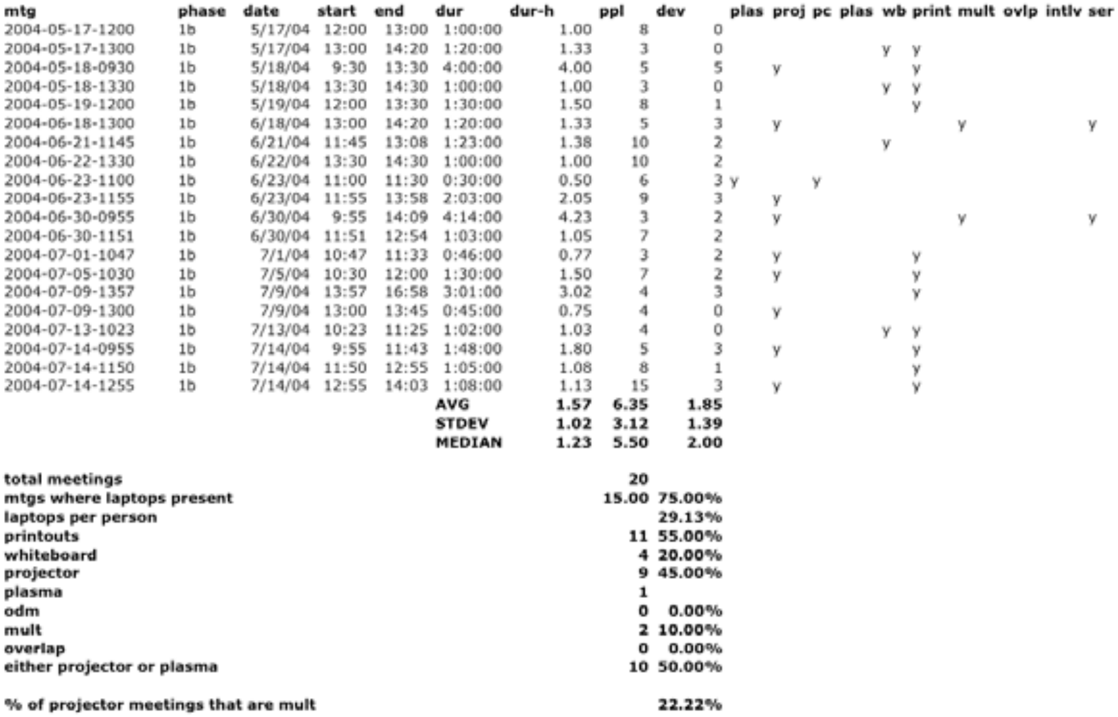
A.1 Phase 1a Meeting Data



mtgname	phase	start	end	dur	dur-h	ppl	dev	used	print	wb	proj	odm	mult	ovlp	intlv	ser
2004-02-03-1031	1a	10:31	12:08	1:37:00	1.62	6	3	3	n	n	y	n	2	0	0	2
2004-02-04-1158	1a	11:58	13:28	1:30:00	1.50	7	0	0	y	n	n	n	0	0	0	0
2004-02-06-1225	1a	12:25	16:11	3:46:00	3.77	5	3	3	n	y	y	n	2	0	0	2
2004-02-09-1158	1a	11:58	13:41	1:43:00	1.72	8	0	0	n	n	n	n	0	0	0	0
2004-02-10-1033	1a	10:33	12:06	1:33:00	1.55	4	1	1	n	n	y	n	0	0	0	0
2004-02-11-1154	1a	11:54	13:00	1:06:00	1.10	7	0	0	y	n	n	n	0	0	0	0
2004-02-16-1154	1a	11:54	12:44	0:50:00	0.83	8	1	1	y	n	n	n	0	0	0	0
2004-02-17-1030	1a	10:30	11:16	0:46:00	0.77	2	2	2	n	n	n	n	0	0	0	0
2004-02-17-1325	1a	13:25	14:41	1:16:00	1.27	6	3	2	n	n	y	n	0	0	0	0
2004-02-18-1059	1a	10:59	11:18	0:19:00	0.32	4	2	1	n	n	n	n	0	0	0	0
2004-02-19-1155	1a	11:55	13:08	1:13:00	1.22	7	0	0	y	n	n	n	0	0	0	0
2004-02-19-1429	1a	14:29	14:59	0:30:00	0.50	6	1	0	n	n	n	n	0	0	0	0
2004-02-23-1001	1a	10:01	11:17	1:16:00	1.27	5	5	4	n	n	y	n	3	0	0	3
2004-02-23-1159	1a	11:59	13:02	1:03:00	1.05	7	0	0	n	n	n	n	0	0	0	0
2004-02-24-1020	1a	10:20	11:10	0:50:00	0.83	2	0	0	y	y	n	n	0	0	0	0
2004-02-25-1156	1a	11:56	13:09	1:13:00	1.22	6	0	0	y	n	n	n	0	0	0	0
2004-03-01-1154	1a	11:54	13:00	1:06:00	1.10	10	1	1	n	y	y	n	0	0	0	0
2004-03-02-1029	1a	10:29	11:53	1:24:00	1.40	4	2	1	n	n	y	n	0	0	0	0
2004-03-02-1425	1a	14:25	14:39	0:14:00	0.23	4	1	0	n	n	n	n	0	0	0	0
2004-03-03-1100	1a	11:00	11:15	0:15:00	0.25	7	2	2	n	n	n	n	0	0	0	0
2004-03-03-1328	1a	13:28	14:15	0:47:00	0.78	6	0	0	y	n	n	n	0	0	0	0
2004-03-24-1257	1a	12:57	13:56	0:59:00	0.98	7	1	1	y	n	n	n	0	0	0	0
2004-03-24-1624	1a	16:24	17:04	0:40:00	0.67	3	1	1	n	n	n	n	0	0	0	0
2004-03-26-1057	1a	10:57	12:04	1:07:00	1.12	6	3	3	y	n	y	n	0	0	0	0
2004-04-01-0958	1a	9:58	11:52	1:54:00	1.90	5	4	3	n	n	y	n	0	0	0	0
2004-04-02-0956	1a	9:56	11:29	1:33:00	1.55	5	1	1	y	n	y	n	0	0	0	0
2004-04-02-1345	1a	13:45	15:04	1:19:00	1.32	10	3	2	y	n	y	n	0	0	0	0
			AVG		**1.18**	**5.81**	**1.48**	**1.19**								
			STDEV		**0.68**	**2.02**	**1.40**	**1.21**								
			MEDIAN		**1.12**	**6.00**	**1.00**	**1.00**								

total meetings	**27**	
mtgs where laptops present	**19.00**	**70.37%**
mtgs where laptops used	**17.00**	**62.96%**
laptops per person		**25.48%**
laptops used per person		**20.41%**
printouts	**11**	**40.74%**
whiteboard	**3**	**11.11%**
projector	**11**	**40.74%**
odm	**0**	**0.00%**
mult	**3**	**11.11%**
overlap	**0**	**0.00%**
% of projector meetings that are mult		**27.27%**

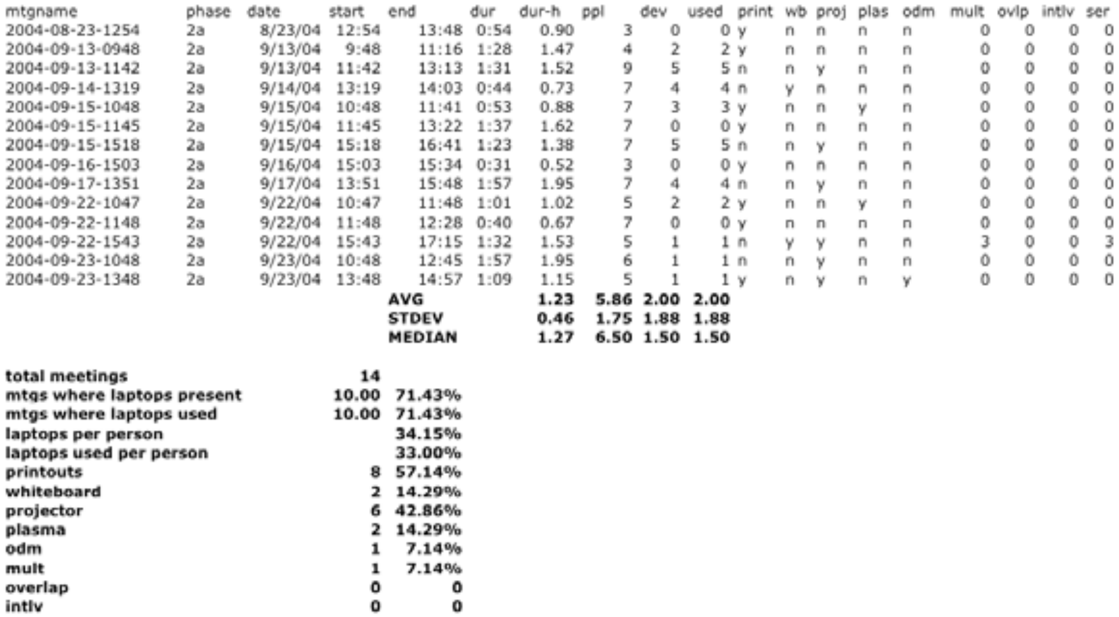
A.2 Phase 1b Meeting Data



mtg	phase	date	start	end	dur	dur-h	ppl	dev	plas	proj	pc	plas	wb	print	mult	ovlp	intlv	ser
2004-05-17-1200	1b	5/17/04	12:00	13:00	1:00:00	1.00	8	0										
2004-05-17-1300	1b	5/17/04	13:00	14:20	1:20:00	1.33	3	0					y	y				
2004-05-18-0930	1b	5/18/04	9:30	13:30	4:00:00	4.00	5	5		y				y				
2004-05-18-1330	1b	5/18/04	13:30	14:30	1:00:00	1.00	3	0					y	y				
2004-05-19-1200	1b	5/19/04	12:00	13:30	1:30:00	1.50	8	1						y				
2004-06-18-1300	1b	6/18/04	13:00	14:20	1:20:00	1.33	5	3		y					y			y
2004-06-21-1145	1b	6/21/04	11:45	13:08	1:23:00	1.38	10	2					y					
2004-06-22-1330	1b	6/22/04	13:30	14:30	1:00:00	1.00	10	2										
2004-06-23-1100	1b	6/23/04	11:00	11:30	0:30:00	0.50	6	3	y		y							
2004-06-23-1155	1b	6/23/04	11:55	13:58	2:03:00	2.05	9	3		y								
2004-06-30-0955	1b	6/30/04	9:55	14:09	4:14:00	4.23	3	2		y					y			y
2004-06-30-1151	1b	6/30/04	11:51	12:54	1:03:00	1.05	7	2										
2004-07-01-1047	1b	7/1/04	10:47	11:33	0:46:00	0.77	3	2		y				y				
2004-07-05-1030	1b	7/5/04	10:30	12:00	1:30:00	1.50	7	2		y				y				
2004-07-09-1357	1b	7/9/04	13:57	16:58	3:01:00	3.02	4	3						y				
2004-07-09-1300	1b	7/9/04	13:00	13:45	0:45:00	0.75	4	0		y								
2004-07-13-1023	1b	7/13/04	10:23	11:25	1:02:00	1.03	4	0					y	y				
2004-07-14-0955	1b	7/14/04	9:55	11:43	1:48:00	1.80	5	3		y				y				
2004-07-14-1150	1b	7/14/04	11:50	12:55	1:05:00	1.08	8	1						y				
2004-07-14-1255	1b	7/14/04	12:55	14:03	1:08:00	1.13	15	3		y				y				
					AVG	**1.57**	**6.35**	**1.85**										
					STDEV	**1.02**	**3.12**	**1.39**										
					MEDIAN	**1.23**	**5.50**	**2.00**										

total meetings	**20**	
mtgs where laptops present	**15.00**	**75.00%**
laptops per person		**29.13%**
printouts	**11**	**55.00%**
whiteboard	**4**	**20.00%**
projector	**9**	**45.00%**
plasma	**1**	
odm	**0**	**0.00%**
mult	**2**	**10.00%**
overlap	**0**	**0.00%**
either projector or plasma	**10**	**50.00%**
% of projector meetings that are mult		**22.22%**

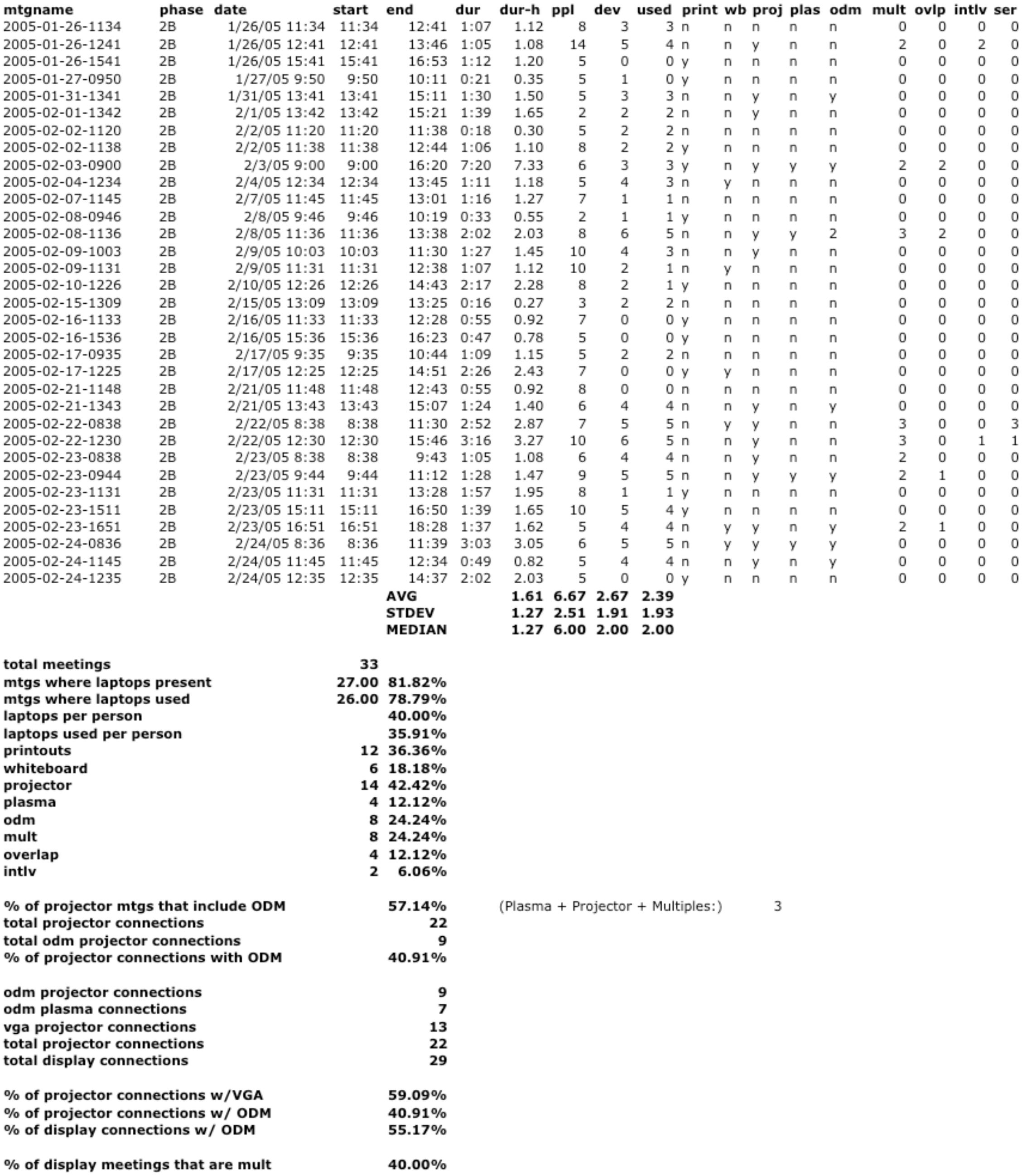
A.3 Phase 2a Meeting Data



mtgname	phase	date	start	end	dur	dur-h	ppl	dev	used	print	wb	proj	plas	odm	mult	ovlp	intlv	ser
2004-08-23-1254	2a	8/23/04	12:54	13:48	0:54	0.90	3	0	0	y	n	n	n	n	0	0	0	0
2004-09-13-0948	2a	9/13/04	9:48	11:16	1:28	1.47	4	2	2	y	n	n	n	n	0	0	0	0
2004-09-13-1142	2a	9/13/04	11:42	13:13	1:31	1.52	9	5	5	n	n	y	n	n	0	0	0	0
2004-09-14-1319	2a	9/14/04	13:19	14:03	0:44	0.73	7	4	4	n	y	n	n	n	0	0	0	0
2004-09-15-1048	2a	9/15/04	10:48	11:41	0:53	0.88	7	3	3	y	n	n	y	n	0	0	0	0
2004-09-15-1145	2a	9/15/04	11:45	13:22	1:37	1.62	7	0	0	y	n	n	n	n	0	0	0	0
2004-09-15-1518	2a	9/15/04	15:18	16:41	1:23	1.38	7	5	5	n	n	y	n	n	0	0	0	0
2004-09-16-1503	2a	9/16/04	15:03	15:34	0:31	0.52	3	0	0	y	n	n	n	n	0	0	0	0
2004-09-17-1351	2a	9/17/04	13:51	15:48	1:57	1.95	7	4	4	n	n	y	n	n	0	0	0	0
2004-09-22-1047	2a	9/22/04	10:47	11:48	1:01	1.02	5	2	2	y	n	n	y	n	0	0	0	0
2004-09-22-1148	2a	9/22/04	11:48	12:28	0:40	0.67	7	0	0	y	n	n	n	n	0	0	0	0
2004-09-22-1543	2a	9/22/04	15:43	17:15	1:32	1.53	5	1	1	n	y	y	n	n	3	0	0	3
2004-09-23-1048	2a	9/23/04	10:48	12:45	1:57	1.95	6	1	1	n	n	y	n	n	0	0	0	0
2004-09-23-1348	2a	9/23/04	13:48	14:57	1:09	1.15	5	1	1	y	n	y	n	y	0	0	0	0
				AVG		**1.23**	**5.86**	**2.00**	**2.00**									
				STDEV		**0.46**	**1.75**	**1.88**	**1.88**									
				MEDIAN		**1.27**	**6.50**	**1.50**	**1.50**									

total meetings	**14**	
mtgs where laptops present	**10.00**	**71.43%**
mtgs where laptops used	**10.00**	**71.43%**
laptops per person		**34.15%**
laptops used per person		**33.00%**
printouts	**8**	**57.14%**
whiteboard	**2**	**14.29%**
projector	**6**	**42.86%**
plasma	**2**	**14.29%**
odm	**1**	**7.14%**
mult	**1**	**7.14%**
overlap	**0**	**0**
intlv	**0**	**0**

A.4 Phase 2b Meeting Data



mtgname	phase	date	start	end	dur	dur-h	ppl	dev	used	print	wb	proj	plas	odm	mult	ovlp	intlv	ser
2005-01-26-1134	2B	1/26/05 11:34	11:34	12:41	1:07	1.12	8	3	3	n	n	n	n	n	0	0	0	0
2005-01-26-1241	2B	1/26/05 12:41	12:41	13:46	1:05	1.08	14	5	4	n	n	y	n	n	2	0	2	0
2005-01-26-1541	2B	1/26/05 15:41	15:41	16:53	1:12	1.20	5	0	0	y	n	n	n	n	0	0	0	0
2005-01-27-0950	2B	1/27/05 9:50	9:50	10:11	0:21	0.35	5	1	0	y	n	n	n	n	0	0	0	0
2005-01-31-1341	2B	1/31/05 13:41	13:41	15:11	1:30	1.50	5	3	3	n	n	y	n	y	0	0	0	0
2005-02-01-1342	2B	2/1/05 13:42	13:42	15:21	1:39	1.65	2	2	2	n	n	y	n	n	0	0	0	0
2005-02-02-1120	2B	2/2/05 11:20	11:20	11:38	0:18	0.30	5	2	2	n	n	n	n	n	0	0	0	0
2005-02-02-1138	2B	2/2/05 11:38	11:38	12:44	1:06	1.10	8	2	2	y	n	n	n	n	0	0	0	0
2005-02-03-0900	2B	2/3/05 9:00	9:00	16:20	7:20	7.33	6	3	3	y	n	y	y	y	2	2	0	0
2005-02-04-1234	2B	2/4/05 12:34	12:34	13:45	1:11	1.18	5	4	3	n	y	n	n	n	0	0	0	0
2005-02-07-1145	2B	2/7/05 11:45	11:45	13:01	1:16	1.27	7	1	1	n	n	n	n	n	0	0	0	0
2005-02-08-0946	2B	2/8/05 9:46	9:46	10:19	0:33	0.55	2	1	1	y	n	n	n	n	0	0	0	0
2005-02-08-1136	2B	2/8/05 11:36	11:36	13:38	2:02	2.03	8	6	5	n	n	y	y	2	3	2	0	0
2005-02-09-1003	2B	2/9/05 10:03	10:03	11:30	1:27	1.45	10	4	3	n	n	y	n	n	0	0	0	0
2005-02-09-1131	2B	2/9/05 11:31	11:31	12:38	1:07	1.12	10	2	1	n	y	n	n	n	0	0	0	0
2005-02-10-1226	2B	2/10/05 12:26	12:26	14:43	2:17	2.28	8	2	1	y	n	n	n	n	0	0	0	0
2005-02-15-1309	2B	2/15/05 13:09	13:09	13:25	0:16	0.27	3	2	2	n	n	n	n	n	0	0	0	0
2005-02-16-1133	2B	2/16/05 11:33	11:33	12:28	0:55	0.92	7	0	0	y	n	n	n	n	0	0	0	0
2005-02-16-1536	2B	2/16/05 15:36	15:36	16:23	0:47	0.78	5	0	0	y	n	n	n	n	0	0	0	0
2005-02-17-0935	2B	2/17/05 9:35	9:35	10:44	1:09	1.15	5	2	2	n	n	n	n	n	0	0	0	0
2005-02-17-1225	2B	2/17/05 12:25	12:25	14:51	2:26	2.43	7	0	0	y	y	n	n	n	0	0	0	0
2005-02-21-1148	2B	2/21/05 11:48	11:48	12:43	0:55	0.92	8	0	0	n	n	n	n	n	0	0	0	0
2005-02-21-1343	2B	2/21/05 13:43	13:43	15:07	1:24	1.40	6	4	4	n	n	y	n	y	0	0	0	0
2005-02-22-0838	2B	2/22/05 8:38	8:38	11:30	2:52	2.87	7	5	5	n	y	y	n	n	3	0	0	3
2005-02-22-1230	2B	2/22/05 12:30	12:30	15:46	3:16	3.27	10	6	5	n	n	y	n	n	3	0	1	1
2005-02-23-0838	2B	2/23/05 8:38	8:38	9:43	1:05	1.08	6	4	4	n	n	y	n	n	2	0	0	0
2005-02-23-0944	2B	2/23/05 9:44	9:44	11:12	1:28	1.47	9	5	5	n	n	y	y	y	2	1	0	0
2005-02-23-1131	2B	2/23/05 11:31	11:31	13:28	1:57	1.95	8	1	1	y	n	n	n	n	0	0	0	0
2005-02-23-1511	2B	2/23/05 15:11	15:11	16:50	1:39	1.65	10	5	4	y	n	n	n	n	0	0	0	0
2005-02-23-1651	2B	2/23/05 16:51	16:51	18:28	1:37	1.62	5	4	4	n	y	y	n	y	2	1	0	0
2005-02-24-0836	2B	2/24/05 8:36	8:36	11:39	3:03	3.05	6	5	5	n	y	y	y	y	0	0	0	0
2005-02-24-1145	2B	2/24/05 11:45	11:45	12:34	0:49	0.82	5	4	4	n	n	y	n	y	0	0	0	0
2005-02-24-1235	2B	2/24/05 12:35	12:35	14:37	2:02	2.03	5	0	0	y	n	n	n	n	0	0	0	0
				AVG		**1.61**	**6.67**	**2.67**	**2.39**									
				STDEV		**1.27**	**2.51**	**1.91**	**1.93**									
				MEDIAN		**1.27**	**6.00**	**2.00**	**2.00**									

total meetings	**33**	
mtgs where laptops present	**27.00**	**81.82%**
mtgs where laptops used	**26.00**	**78.79%**
laptops per person		**40.00%**
laptops used per person		**35.91%**
printouts	**12**	**36.36%**
whiteboard	**6**	**18.18%**
projector	**14**	**42.42%**
plasma	**4**	**12.12%**
odm	**8**	**24.24%**
mult	**8**	**24.24%**
overlap	**4**	**12.12%**
intlv	**2**	**6.06%**

% of projector mtgs that include ODM	**57.14%**	(Plasma + Projector + Multiples:)	3
total projector connections	**22**		
total odm projector connections	**9**		
% of projector connections with ODM	**40.91%**		

odm projector connections	**9**
odm plasma connections	**7**
vga projector connections	**13**
total projector connections	**22**
total display connections	**29**

% of projector connections w/VGA	**59.09%**
% of projector connections w/ ODM	**40.91%**
% of display connections w/ ODM	**55.17%**

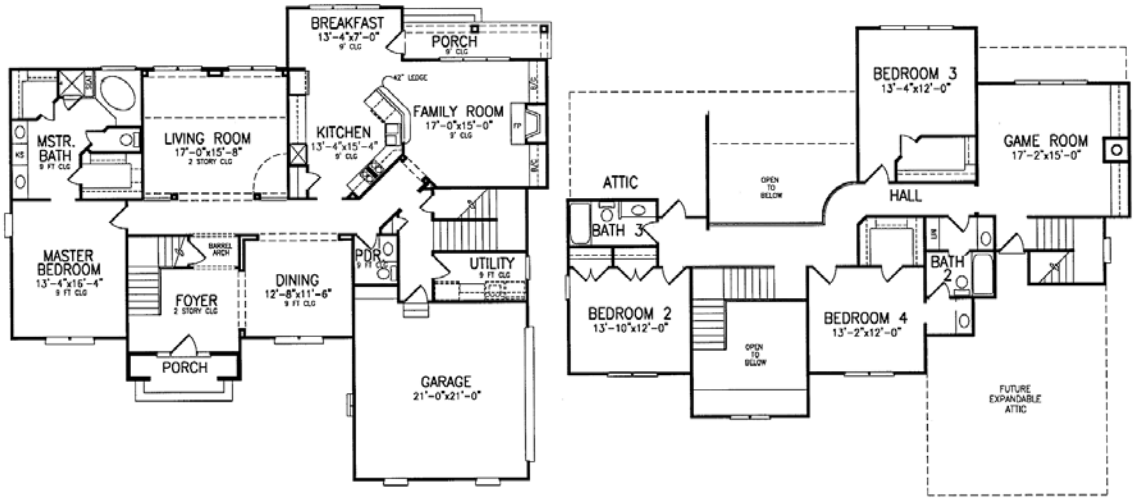
% of display meetings that are mult	**40.00%**

B OSCAR Household Personas

Here are four households that we believe could benefit from the MonsterPad. Each is characterized by significant media consumption and at least one household member with the interest in and capability to customize their media environment to some extent.

B.1 The Engstroms, Maple Grove, MN

House: 2-story, 6 bedroom house in a recently-built suburban community



B.1.1 Household Members:

- Dan, 52, small business owner (auto parts distributor)
- Michelle, 50, schoolteacher
- Audrey, 17, high school junior
- Bridgette, 14, 8th grader
- Duncan, 11, 5th grader
- Butch, 3, golden lab

Dan is an enthusiastic consumer of A/V equipment, which he primarily uses to watch sports and movies. He also makes home movies and shares them with his many nearby family members (mother, father, and 3 siblings).

Michelle teaches 10th grade English at Maple Grove High. She considers herself to be a "low-tech person." She enjoys watching TV and movies, and listening to the radio. Her favorite TV shows are cooking shows and things like "Trading Spaces" or "What Not To Wear."

Audrey is a popular teenager and spends a lot of time cultivating her friendships via phone, SMS, IM, email, and in person. She has her own car.

Bridgette is sullen and spends a lot of time listening to music in her room or on her headphones. She also spends a lot of time in online chat rooms with other sullen teens.

Duncan likes to play video games and watch TV.

B.1.2 How they use media:

Dan:

- Family movie night
- Packers/Vikings games
- Sports in garage, living room, bedroom
- Movies in livingroom, bedroom
- Music and radio in garage, car
- Showing home movies & photos to visitors

Michelle:

- Calling mom, family members while cooking (hands free)
- Leave reminders for kids, Dan
- Watch TV in kitchen, living room, bedroom
- Listen to radio in kitchen, living room, bedroom
- Occasionally recorded music in kitchen, car
- Updates the pictures in the picture frames all over the house

Audrey:

- Mostly uses her cellphone for calls, SMS; laptop for IM
- Watches TV and movies in her bedroom, sometimes in livingroom
- Listens to music in bedroom, headphones, car

Bridgette:

- Uses computer in her room for chat, web surfing
- Listens to music in headphones, room (never in living room!)
- Sometimes Audrey and Bridgette share their music collections with each other
- Watches TV and movies in her bedroom

Duncan:

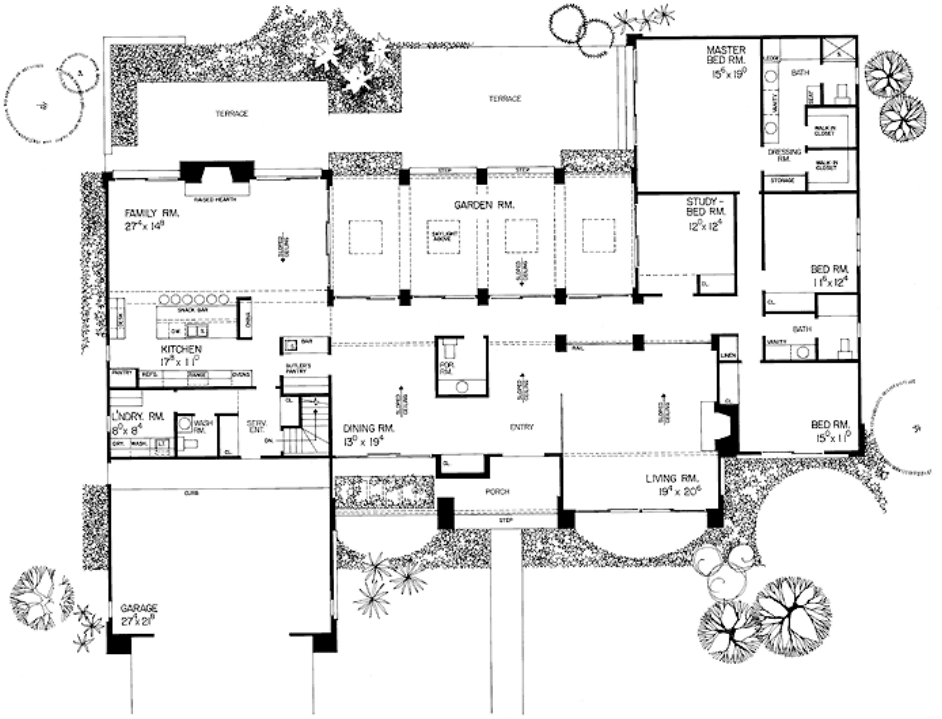
- Plays video games in living room, game room
- Watches TV, movies in living room, game room, bedroom
- Listens to radio and music in living room, bedroom

Misc:

- Dan and Michelle have dinner parties, play music at them
- At Christmas and other special occasions, they set up a video link with Michelle's mother and father, who live in Florida
- Dan takes lots of videos and photos of family vacations, and also the kids sporting and other events (Audrey: volleyball and softball, Bridgette: cross country, school plays and band, Duncan: soccer and tae kwon do). He maintains a library of these media to show off later.

B.2 The Merriweather-Alvarez household, Sedona, AZ

House: 1-story, 3-bedroom house on 2 acres outside of town



B.2.1 Household Members:

- Haley Merriweather, 37, massage therapist
- Angie Alvarez, 43, web project manager
- Mario, 13 and Luigi, 11: cats

Haley is a 37-year old massage therapist, her partner is **Angie** who is a web project management contractor.

They both work at home, though Angie's job takes her outside the house and even out of town on a regular basis.

Haley's aging mother lives in Dayton, Ohio and Haley makes a great deal of effort to stay in touch.

Angie is fascinated with new gear and is often bringing home new gear and changing stuff around in the household. Haley leaves most of the system configuration and buying decisions to Angie but is very annoyed when things don't work for her. Primarily, Haley watches movies and listens to music--including the New Age music that she plays in the massage studio. They take a lot of digital photographs in their travels around the desert and maintain a website to showcase their photos.

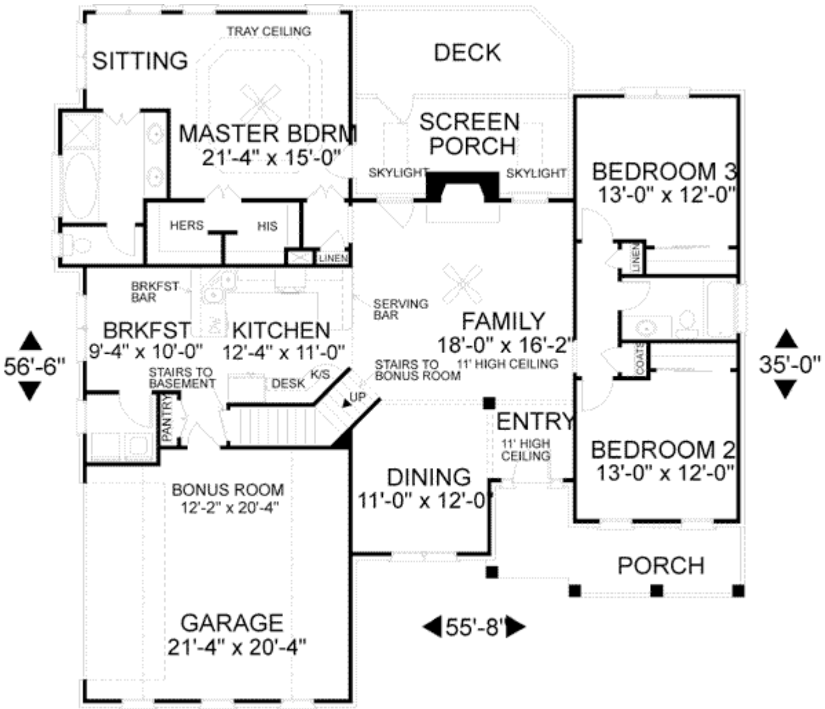
B.2.2 How they use media

- Angie maintains the family website, to which she posts lots of photos of their travels and blog entries which tend to focus around the activities of the cats.
- Haley and Angie watch lots of movies together in the livingroom and bedroom.
- Angie likes to listen to music while working in the kitchen or around the house.
- Haley likes to watch TV while in the kitchen or around the house.
- They both like to listen to music and drink wine on the back deck in the evenings.
- They have friends over for dinner somewhat regularly, for which they often play background music.
- Haley uses one of the spare bedrooms as her massage studio. She has a special collection of relaxing music that she plays when giving clients massages. She also dims the lights and projects slowly moving clouds on the ceiling.
- Angie likes to show Haley funny or bizarre stuff that she's come across on the Web.
- They both take lots of pictures with their digital cameras.

- Angie listens to music while working in her home office.
- Haley tries to call her mother every day or two, but sometimes just sends a message to her mom with some news of the day. She often likes to call while busy with housework or while doing some menial task in her massage studio.
- The leave notes and reminders for each other. This is usually paper, but sometimes includes a message on the answering machine.

B.3 Kari, Jesse, and Ruth, Durham, NC

3-bedroom house near Duke University



B.3.1 Household member:

- Kari Weissbrun, 26, magazine editor
- Jesse Amman, 26, waiter
- Ruth Sloan, 28, video artist/educator

Kari spends a significant amount of her free time playing World of Warcraft and other online games. She also makes frequent use of electronic communication technologies (email, IM, blogging, chat) to maintain contact with her college friends who have scattered to various cities throughout the US. She and her roommates download a lot of music and video from the Internet. They watch lots of movies and listen to music pretty much constantly. Kari is a heavy computer and Internet user but does not know how to program or write scripts at all.

Jesse also plays a lot of MMORPG, often connecting online with Holly while they are each sitting in their separate bedrooms. He is an aspiring laptop DJ and spends a lot of time researching, collecting, and organizing music.

Ruth is a video artist and has a somewhat sophisticated digital video editing setup in her bedroom. She doesn't hang out with Holly and Jesse all that much, though occasionally they will all eat dinner and watch a movie together. She tends to hang out in her bedroom when she's home, where she works on video and other projects, listens to music, and watches movies. Her boyfriend Rafael is over a lot, and they mainly watch movies in Ruth's bedroom.

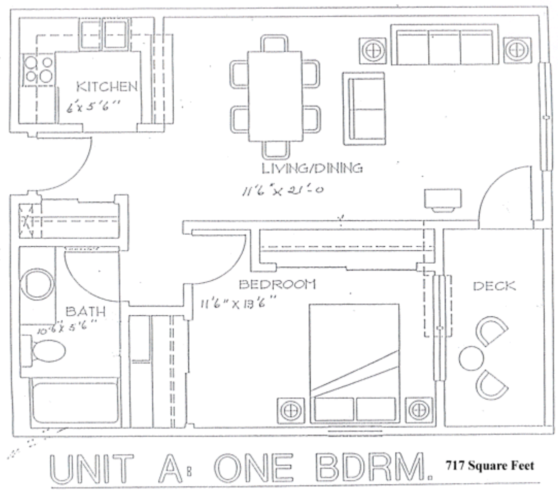
B.3.2 How they use media

- All of them listen to music in their rooms, the living room, the kitchen, the bathroom—everywhere. Each has their own music collection but they are all shared with each other.
- They all watch movies and TV, together and alone, all over the house.
- Kari and Jesse share stuff they come across on the web.
- Notes and reminders: chore wheel and bill reminders.

- Kari and Jesse are all into the blogosphere—they each have blogs and spend a lot of effort keeping up with what's going on with several of their friends' and some minor celebrities' blogs.
- Ruth maintains a website for her video art projects.
- She has a server for
- They have parties every few months at which they (mainly Jesse) play music and show videos.

B.4 Eugene An, Seattle, WA

1-bedroom apartment in Capitol Hill



B.4.1 Household members

- Eugene An, 29, musician/administrative assistant
- (Frequent guest: Sabrina Lin, 27, social worker—his girlfriend)

Eugene is a 29 year old part time administrative assistant in the University of Washington Biology Department. He is also a musician who uses computers to create and record his music. He authors a podcast in which he plays uncopyrighted music that he has downloaded from the Internet. While not a system administrator by profession or training, he is the person in his office that colleagues tend to turn to when they have problems with their computers, applications, or peripherals. He lives alone but ends up acting as the "sys admin" for many of his friends and family members, e.g. his father and mother who live in Ashland, OR and who he visits at least once every other month or so.

His girlfriend, **Sabrina**, spends a good deal of time at his apartment, typically spending 1 or 2 nights a week. When she is over, they often watch movies or TV.

B.4.2 How they use media

- Eugene has a very large music collection and listens to music nearly constantly when he is home. He listens to music in all rooms of the house, and when he goes out he listens to music on his portable MP3 player.
- Eugene makes mixes for Sabrina and for his friends.
- He records snippets of various kinds of media (broadcast radio, internet radio, TV, movies) for use in podcasts and electronic music projects.
- He downloads a lot of music from the Internet. He has a very elaborate system for categorizing music that he has downloaded and making notes about what he has listened to, what he likes and what he doesn't.
- He also listens to audio from a variety of sources on the net, especially podcasts, and Internet radio stations.
- He manages Sabrina's portable MP3 player for her, putting music on it that he thinks she will like.
- He and Sabrina watch movies together a few times a week.
- Sabrina watches TV when she is over, sometimes while Eugene is working music projects with his headphones on.

- As part of making music, Eugene listens to versions of his music in each room of his house, through his “good” headphones, and on his MP3 player to hear how it sounds through various kinds of speakers.

C OSCAR Expert Review #1 Materials

C.1 Instructions to reviewers

C.1.1 Overview

The Obje Interoperability Framework is intended to make it much easier for devices to work together than is currently possible. The application we are asking you to evaluate ("MonsterPad") is intended to allow end-users to interact with, control, configure, and *compose* a wide variety of media-oriented devices and services.

The materials being presented to you for review include sketches for a proposed service composition user interface, along with descriptive materials describing the usage setting for which the UI is targeted.

C.1.2 Materials

User Profiles

The users we are targeting with the MonsterPad application are moderate to heavy consumers of media and early adopter to early majority consumers of media technology. They are individuals who have some interest in configuring and improving their media consumption environment, and are somewhat willing to try out new types of devices and technology. They do not necessarily have a high degree of technical skill or knowledge of technical topics such as computer programming, system administration, electrical wiring, or electronics design. They are likely to be comfortable using computers and the Internet but are not necessarily "experts."

Prologue Script

The following script describes the setting in which the MonsterPad is to be used:

You have just purchased a new "MediaMonster" device and brought it home. You bought the device because it promises an "all-in-one solution to all your home media needs, including video-on-demand, Internet music, games, applications, as well as personal audio, video, and photo collections." Since the device is advertised as "Monster-enabled", you know that it will work with your existing home network and media devices (such as speakers and TVs in several rooms, fixed and portable cameras, mobile media devices, and a few others), which are also Monster-enabled.

The MediaMonster comes with a remote control device called the "MonsterPad" that you will use to configure and control your home media network.

You know from your previous experience with Monster-enabled devices that not only can you connect devices within your own home network, but that it is possible to connect to other people's home networks as well (e.g., friends and family members). For some of the *ensuing activities we will assume that you have already established connections between your home network and the networks of certain family members.*

Tasks

#1: Play the "Welcome" audio file on the living room speakers. When finished return to the main screen.

#2: Locate the documentary "Runaway Trains" on the MovieMonster service and begin watching it on the living room TV. When you are done, please return to the main screen.

#3: Create a template that causes the picture frame by the front door to display a new image from your photo library every time you activate the template.

#3a: Now modify the template so that a new photo is displayed every 10 minutes. When you are finished, return to the main screen.

#4: You read in a review for the MediaMonster that it is possible to create a template that causes your music to follow you around the house. Create that template.

#4a: Now modify the template so that your pictures also follow you around the house. When you are done, return to the main screen.

C.1.3 Technical Overview

Here are a few details about how the MonsterPad application and underlying "Monster" environment are architected. Keep in mind, however, that potential end-users may not have access to information about the system architecture.

Assume they will not have any explicit information other than what is revealed through the interface.

Even though the prologue states that the user has had some prior experience with Monster-enabled devices, we are assuming that such prior experience did not include service discovery or composition. The existing Monster-devices were used in more or less conventional ways and only using functionality that was built directly into each device.

Networked Services

In an Obje (a.k.a. "Monster") environment, different types of devices appear on the network as services and they can be easily discovered by client applications such as the MonsterPad. These services are always running and available. The home network on which they are running has sufficient bandwidth to play audio and video without difficulty. Firewall issues having to do with access to the outside world (e.g. "Mom's house") have all been magically solved.

Services and devices can be connected together in flexible, but not completely arbitrary ways. There are three main types of "components" (i.e. devices and services) in the Obje framework: DataSources, DataSinks, and Aggregates. DataSources can be connected to DataSinks as long as there is a compatible data type that they can agree upon (one of the primary advantages of Obje is that it is much easier to find agreeable data types than it is in existing networks).

Aggregates are collections of other components—that is they can contain collections of DataSources, DataSinks, and other Aggregates. A typical Aggregate would be a media library such as a Music Library or Photo Library. Such a component would itself be an Aggregate and a DataSink (because you could add media to the Library by connecting a compatible DataSource to it). It would contain Folders or Albums, which would also be DataSinks and Aggregates. Folders would contain Files/Songs/Photos which would be DataSources. Below is a complete list of all of the components we assumed to exist for the purposes of this prototype.

Templates

A central concept in the MonsterPad application is the "template"—essentially a record that contains information about service compositions. Templates consist of one or more "connections," each of which contains exactly one source slot and exactly one destination slot.

When the template becomes active, each connection slot is filled based on a set of rules that govern that slot. The slot can be filled by user selection or it can be filled automatically. In both cases, there is a query governing the possible contents of the slot. In case of user selection, the query results are presented to the user for selection. In case of automatic selection, one of the query results is selected at random to fill the slot.

The connection itself is also governed by rules. Connections can be made on template activation, on user command, at a particular time, or on a recurring schedule. They can also be made based on external events such as the satisfaction of a query or a change in the state of a device or service. For example, a connection could be triggered by the movement of a device from one room to another, or by the appearance of a particular device on the network.

Status and Control

Connections can be in three states (actually there are more, but only three are visible in the UI):

- unavailable: the connection cannot be made without further configuration
- ready: the connection can be made at the user's request
- active: the connection is currently in progress

When a connection is active, there may be control UIs available for one or more of the slot components.

Templates can also assume a variety of states, which are basically the same as the connection states: unavailable, selectable, and active.

Available Components

The following is the complete set of components we assumed were available when designing the prototype. All components are available on the local network except the MusicMonster and MovieMonster components, which are Internet-based services, and the components listed as belonging to “Mom,” which are located at Mom’s house in a different city.

Component	Type(s)
Cellphone Microphone	DataSource
Dining Room Microphone	DataSource
Dining Room Speakers	DataSink
Dining Room Web Cam	DataSource
Fridge Display	DataSink
Front Door Camera	DataSource
Front Door Picture Frame	DataSink
Headphones	DataSink
Kitchen Speakers	DataSink
Laptop Speakers	DataSink
Living Room Picture Frame	DataSink
Living Room Speakers	DataSink
Living Room TV	DataSink
Mom's Dining Room Picture Frame	DataSink
Mom's Dining Room Speakers	DataSink
Mom's Dining Room Webcam	DataSource
Mom's Phone [Speaker & Microphone]	DataSource, DataSink
Office PC Screen	DataSink
Office Speakers	DataSink
Laptop Files	Aggregate, DataSink
MovieMonster (Internet VoD Service)	Aggregate
MusicMonster (Internet Music Streaming Service)	Aggregate
My Music Library	Aggregate, DataSink
My Photo Library	Aggregate, DataSink
My Video Library	Aggregate, DataSink
Office PC Files	Aggregate, DataSink
Portable Music Player	Aggregate, DataSink

C.1.4 Assessment and Reporting

We would like you produce a written report giving us feedback in each of the following areas. The depth of feedback in each area is up to you and will depend on where you feel the greatest need for improvement lies. The last section is the most important—a prioritized list of the most serious issues overall.

In addition, if appropriate, it would be helpful if you would write a brief overview of the approach(es) and/or techniques (whether formal or informal) you used in reviewing the materials.

Assumptions and Background

Please comment on any weaknesses you find in the assumptions we have made about our target users or the supporting technical infrastructure (e.g. device capabilities, network capabilities, etc.). Also try to highlight any implicit or ambiguous assumptions that you were forced to clarify or make explicit in order to make sense of the interface or experiment.

Profiles and Tasks

Please comment on any weaknesses you find in our characterization of the target users—unrealistic expectations or types of users that have been left out in appropriately.

Also comment on the tasks we have chosen—whether they seem realistic and representative of things that our target users would want to do. If you think of alternative ways of framing these tasks, or of alternative tasks altogether, let us know about them.

User Interface

Of course, we hope the bulk of your feedback will be about the user interface itself.

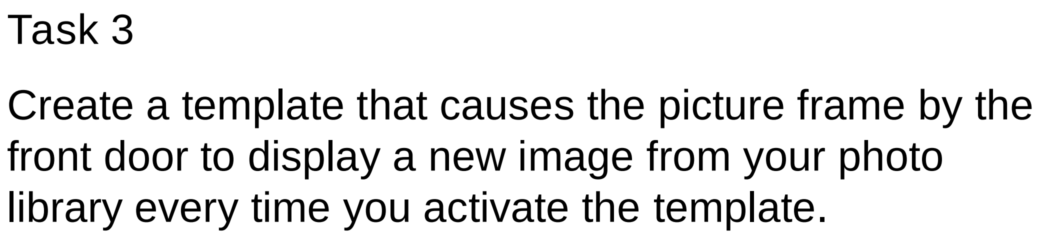
Process Feedback

Any reactions to you have to the process of carrying out this assessment will be most welcome, including the materials provided, the form of these documents, etc.

Summary: Prioritized List of Serious Issues

Please provide a summarized list of the most serious issues and give some sense of their relative severity.

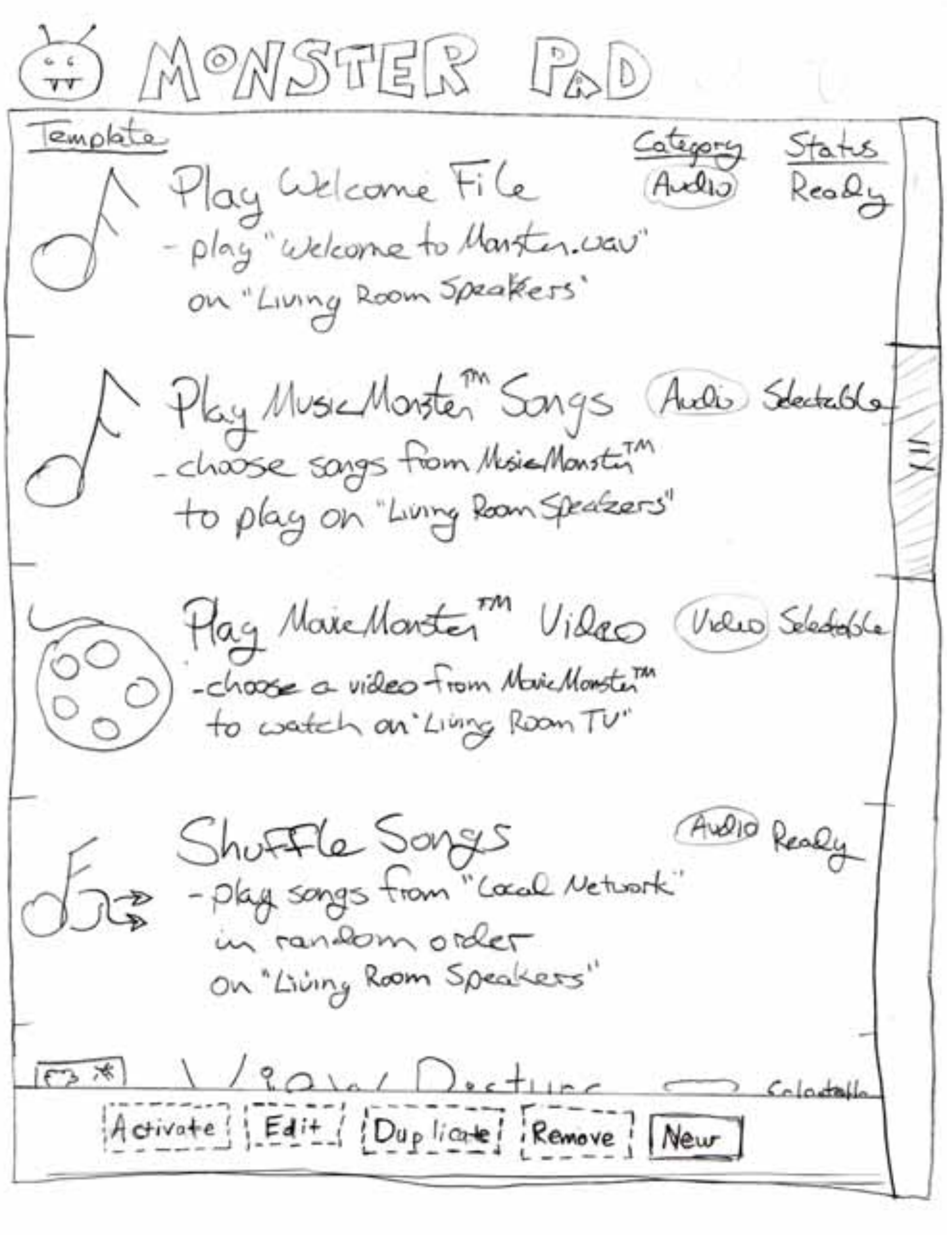
C.2 Example Walkthrough: Task 3



Task 3

Create a template that causes the picture frame by the front door to display a new image from your photo library every time you activate the template.

Task 3, Step 1



Play Welcome File — Audio — Ready
- play "Welcome to Monster.wav"
on "Living Room Speakers"

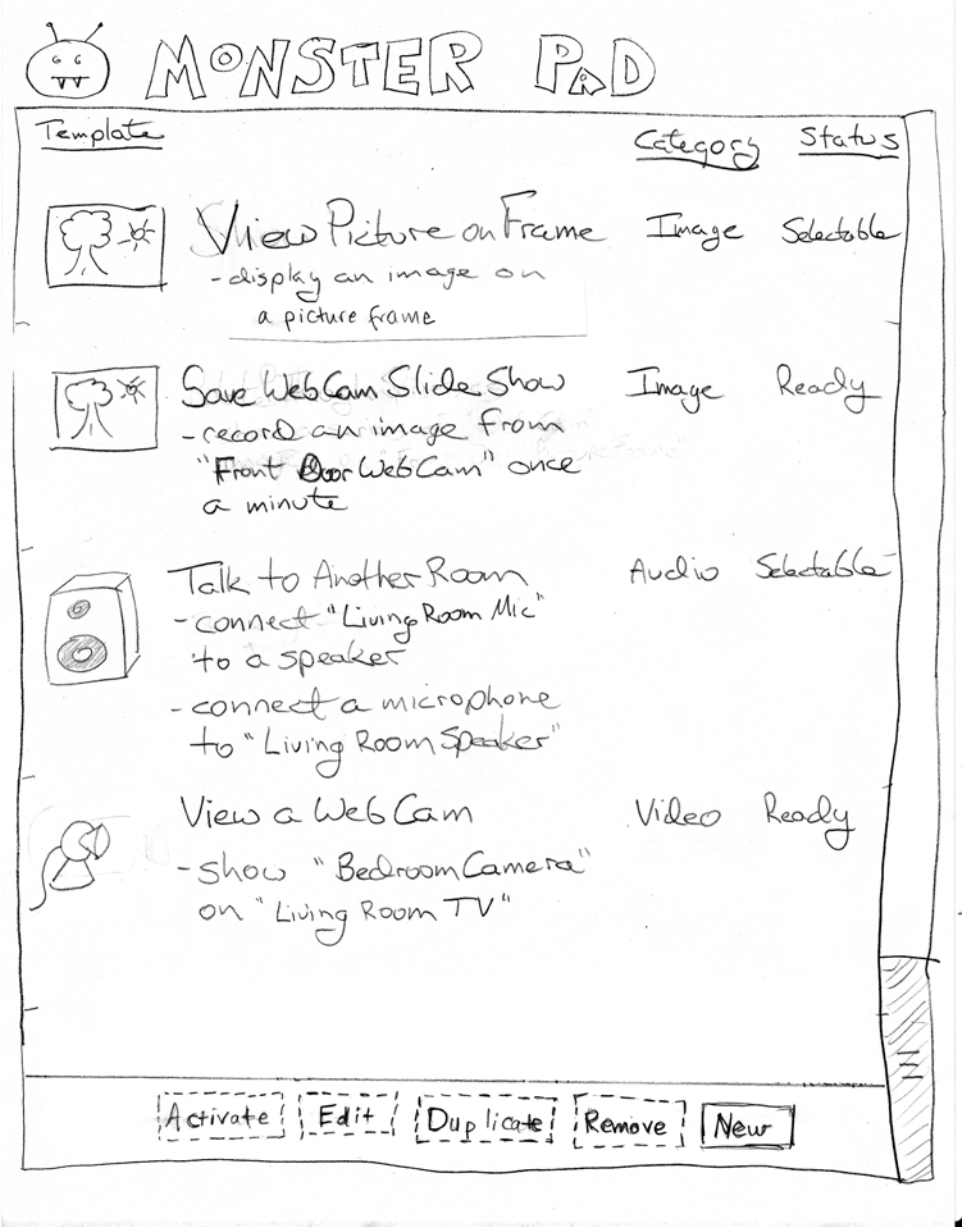
Play MusicMonster™ Songs — Audio — Selectable
- choose songs from MusicMonster™
to play on "Living Room Speakers"

Play MovieMonster™ Video — Video — Selectable
- choose a video from MovieMonster™
to watch on "Living Room TV"

Shuffle Songs — Audio — Ready
- play songs from "Local Network"
in random order
on "Living Room Speakers"

User scrolls down

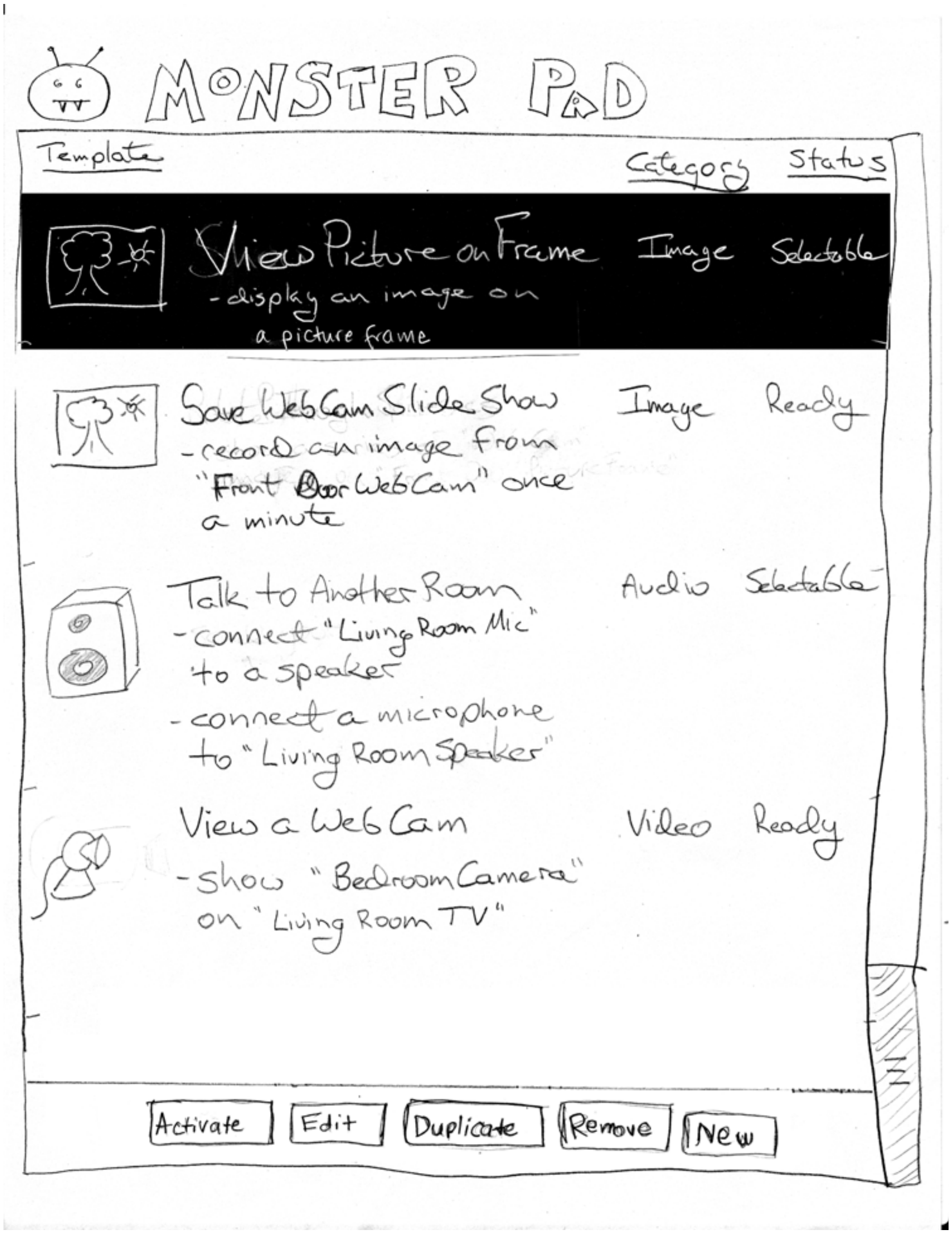
Task 3, Step 2



Template	Category	Status
View Picture on Frame - display an image on a picture frame	Image	Selectable
Save WebCam Slide Show - record an image from "Front Door WebCam" once a minute	Image	Ready
Talk to Another Room - connect "Living Room Mic" to a speaker - connect a microphone to "Living Room Speaker"	Audio	Selectable
View a WebCam - show "Bedroom Camera" on "Living Room TV"	Video	Ready

User selects "View Picture on Frame"

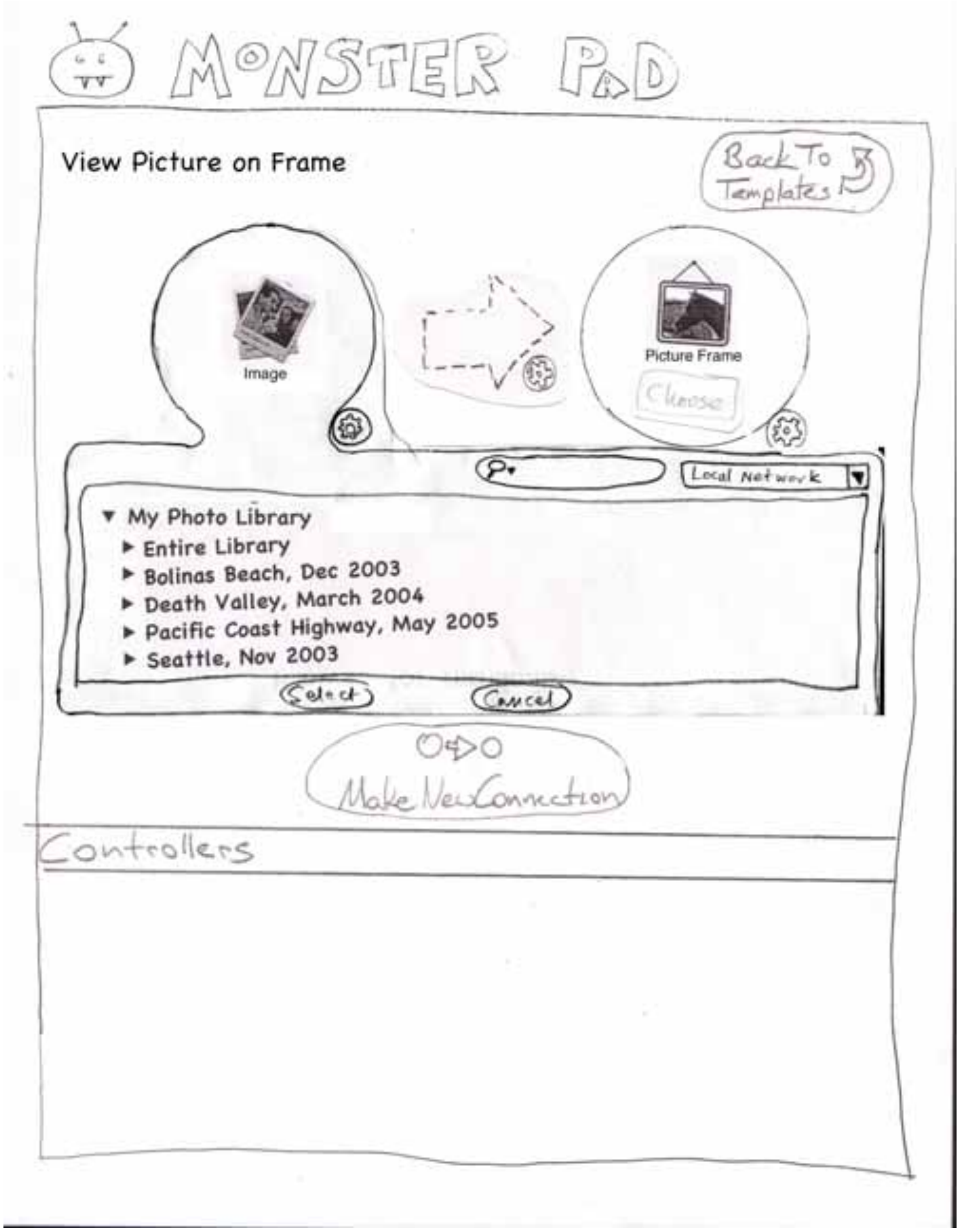
Task 3, Step 3



Template	Category	Status
View Picture on Frame - display an image on a picture frame	Image	Selectable
Save WebCam Slide Show - record an image from "Front Door WebCam" once a minute	Image	Ready
Talk to Another Room - connect "Living Room Mic" to a speaker - connect a microphone to "Living Room Speaker"	Audio	Selectable
View a WebCam - show "Bedroom Camera" on "Living Room TV"	Video	Ready

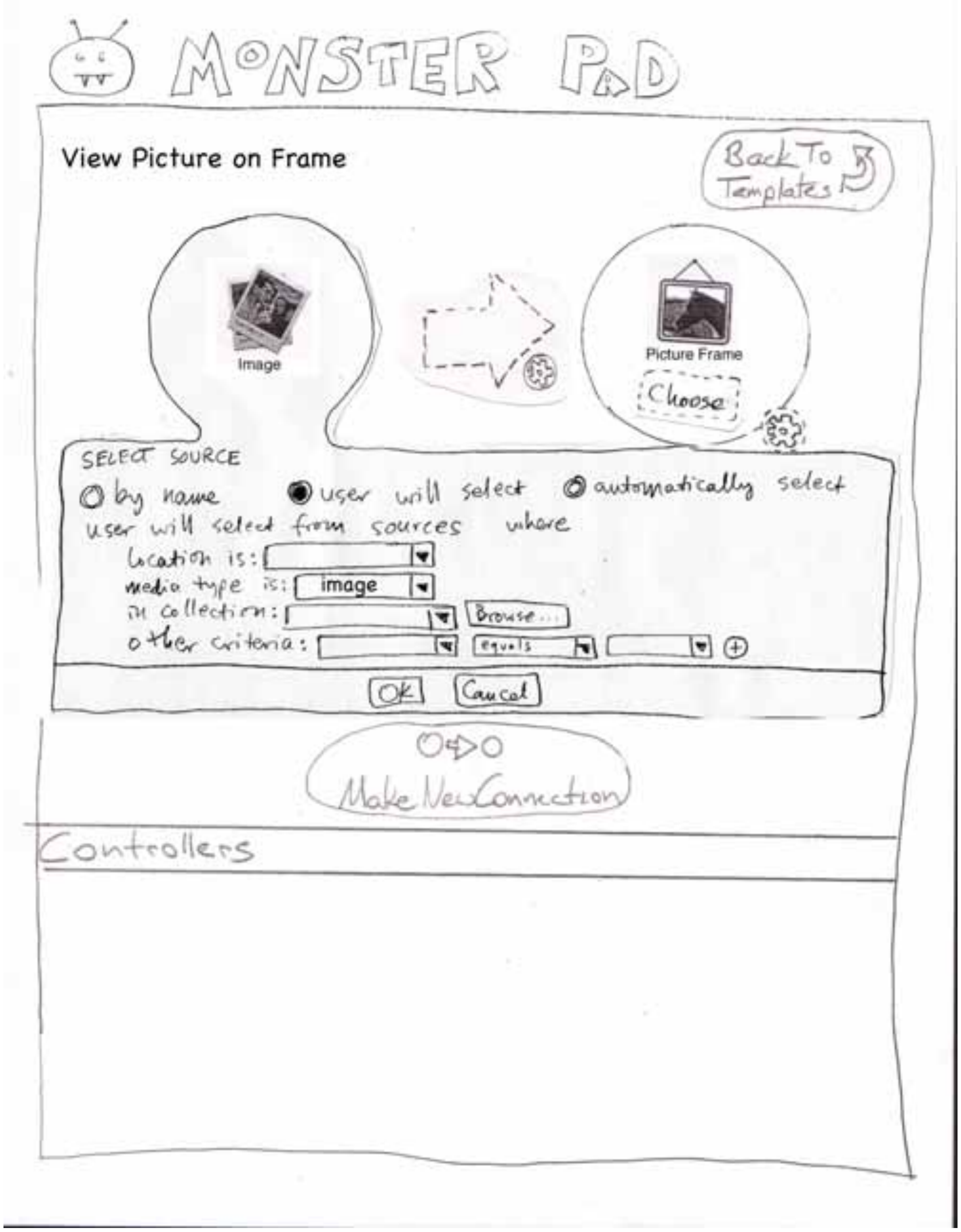
User clicks "Activate"

Task 3, Step 4



User selects the gear icon to the lower right of the Photo/Source slot

Task 3, Step 5



SELECT SOURCE

○ by name ● user will select ○ automatically select

user will select from sources where

location is: ▾

media type is: image ▾

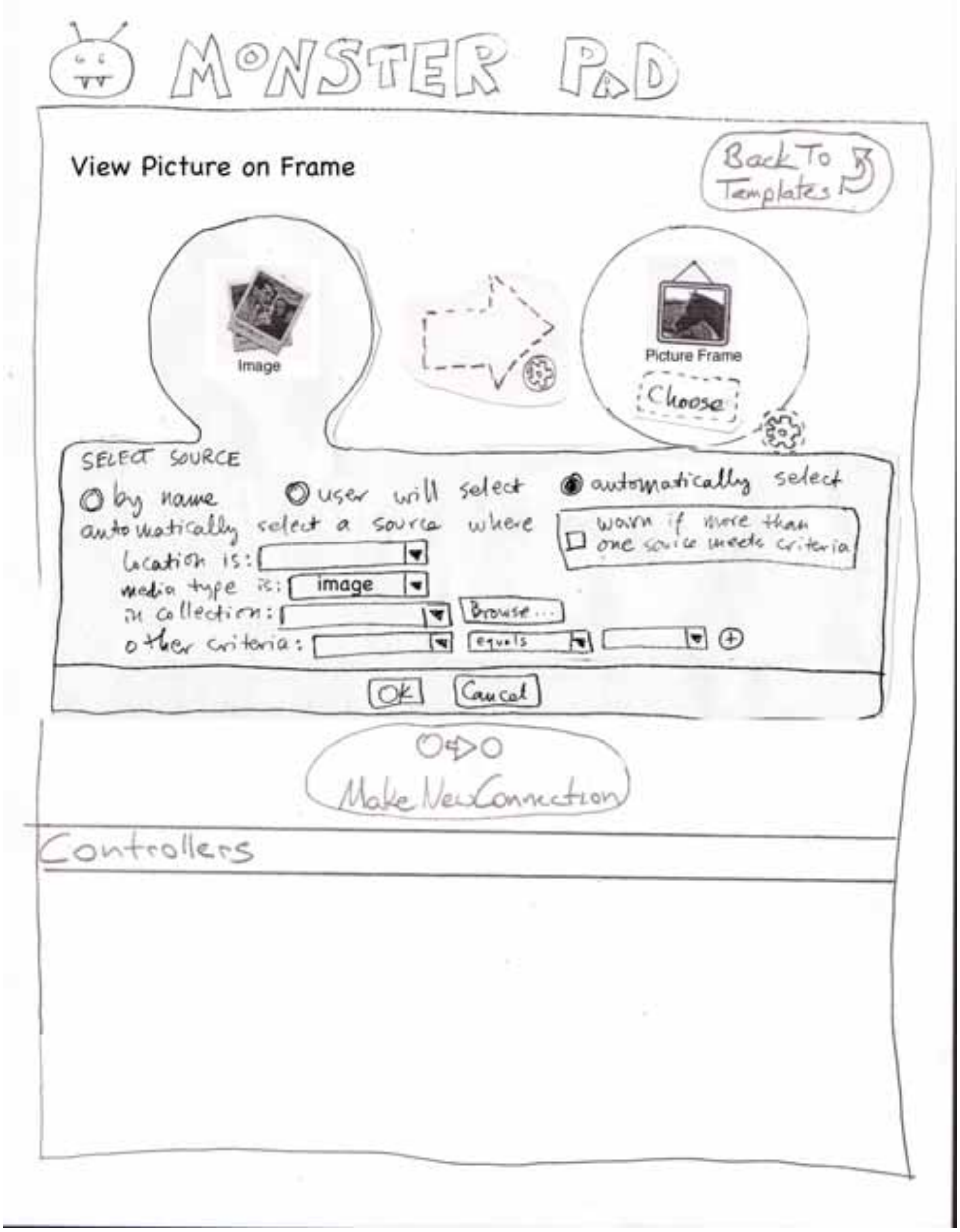
in collection: ▾ Browse...

other criteria: ▾ equals ▾ ▾ ⊕

OK Cancel

User chooses "Automatically Select"

Task 3, Step 6



SELECT SOURCE

○ by name ○ user will select ● automatically select

automatically select a source where

□ warn if more than one source meets criteria

location is: [▾]

media type is: [image ▾]

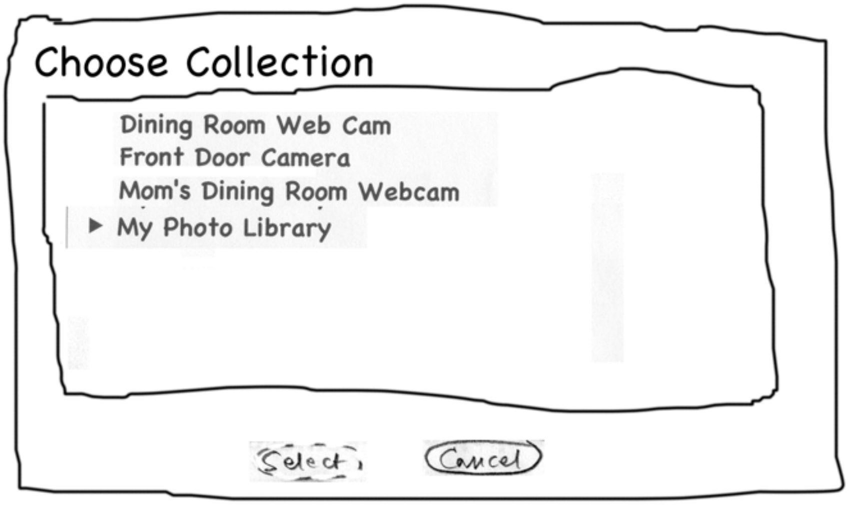
in collection: [▾] Browse...

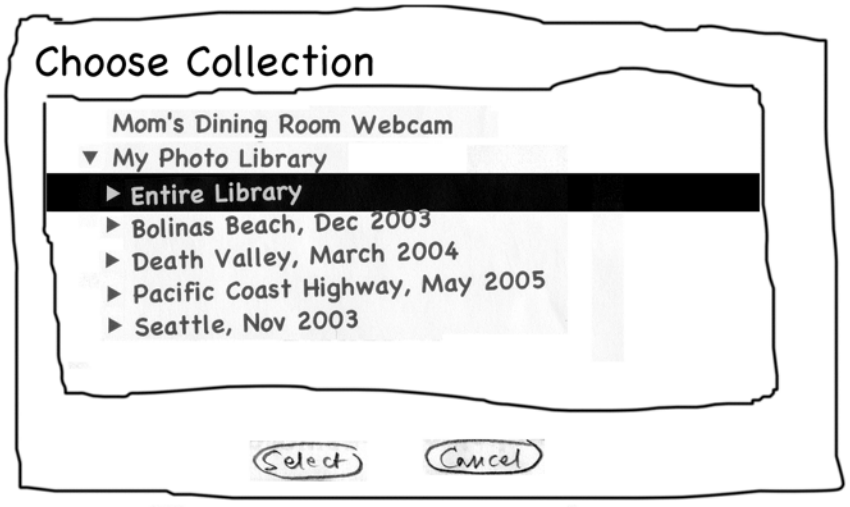
other criteria: [▾] [equals ▾] [▾] ⊕

OK Cancel

User chooses “Browse”

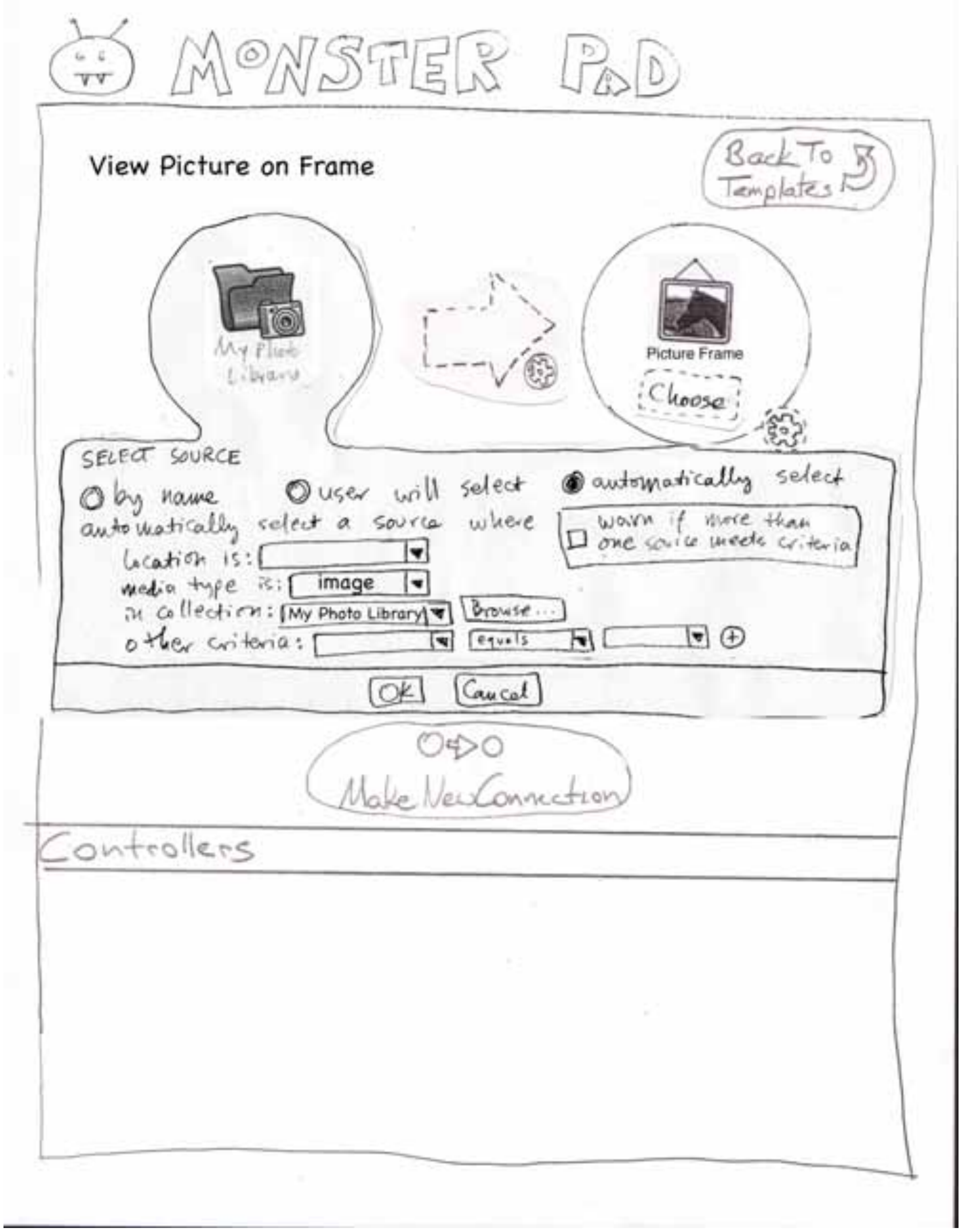
Task 3, Step 7





User hits triangle to expand, then selects "Entire Library", then clicks "Select"

Task 3, Step 8



View Picture on Frame

SELECT SOURCE

○ by name ○ user will select ● automatically select

automatically select a source where

□ warn if more than one source meets criteria

location is: [▾]

media type is: [image ▾]

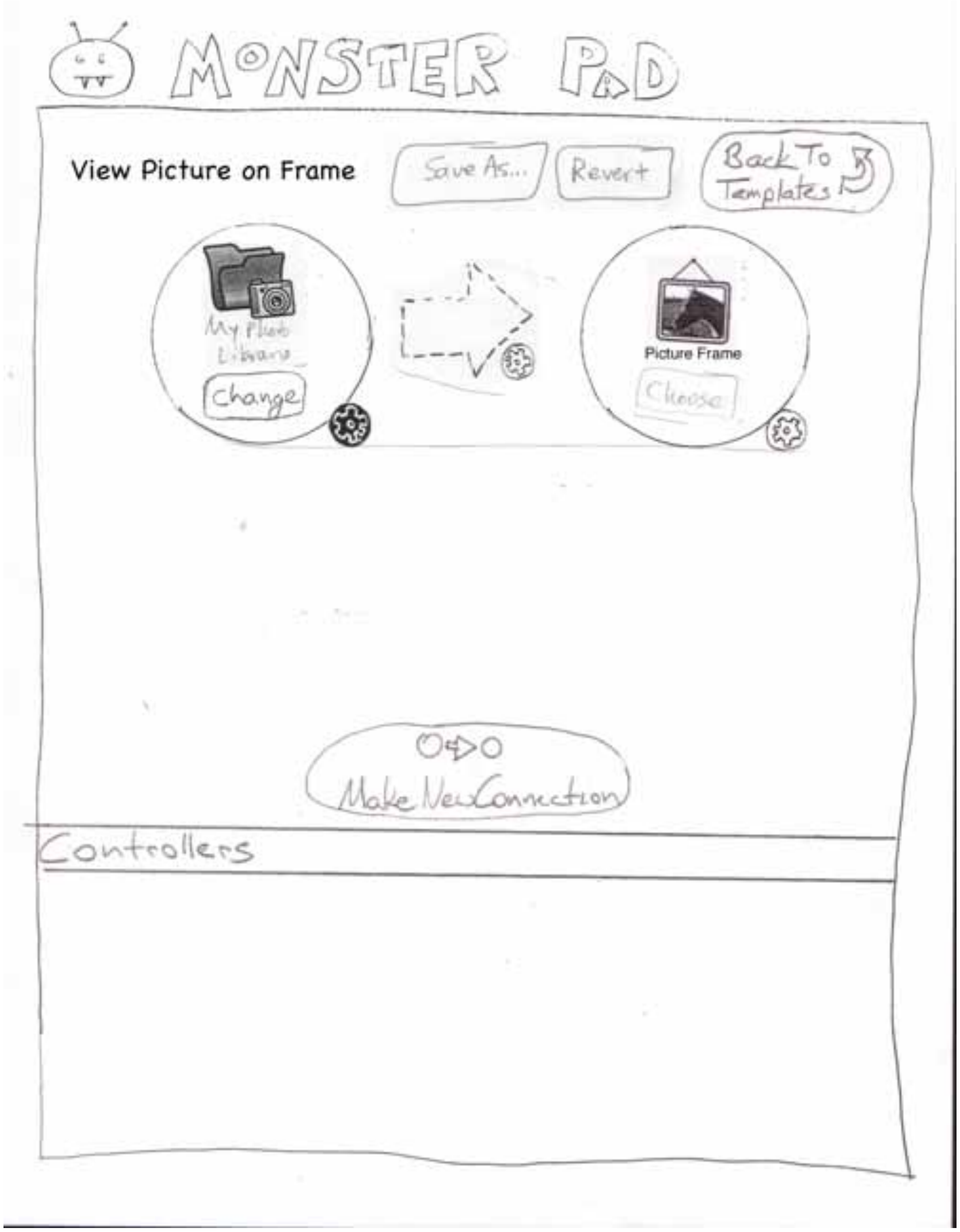
in collection: [My Photo Library ▾] [Browse...]

other criteria: [▾] [equals ▾] [▾] ⊕

[OK] [Cancel]

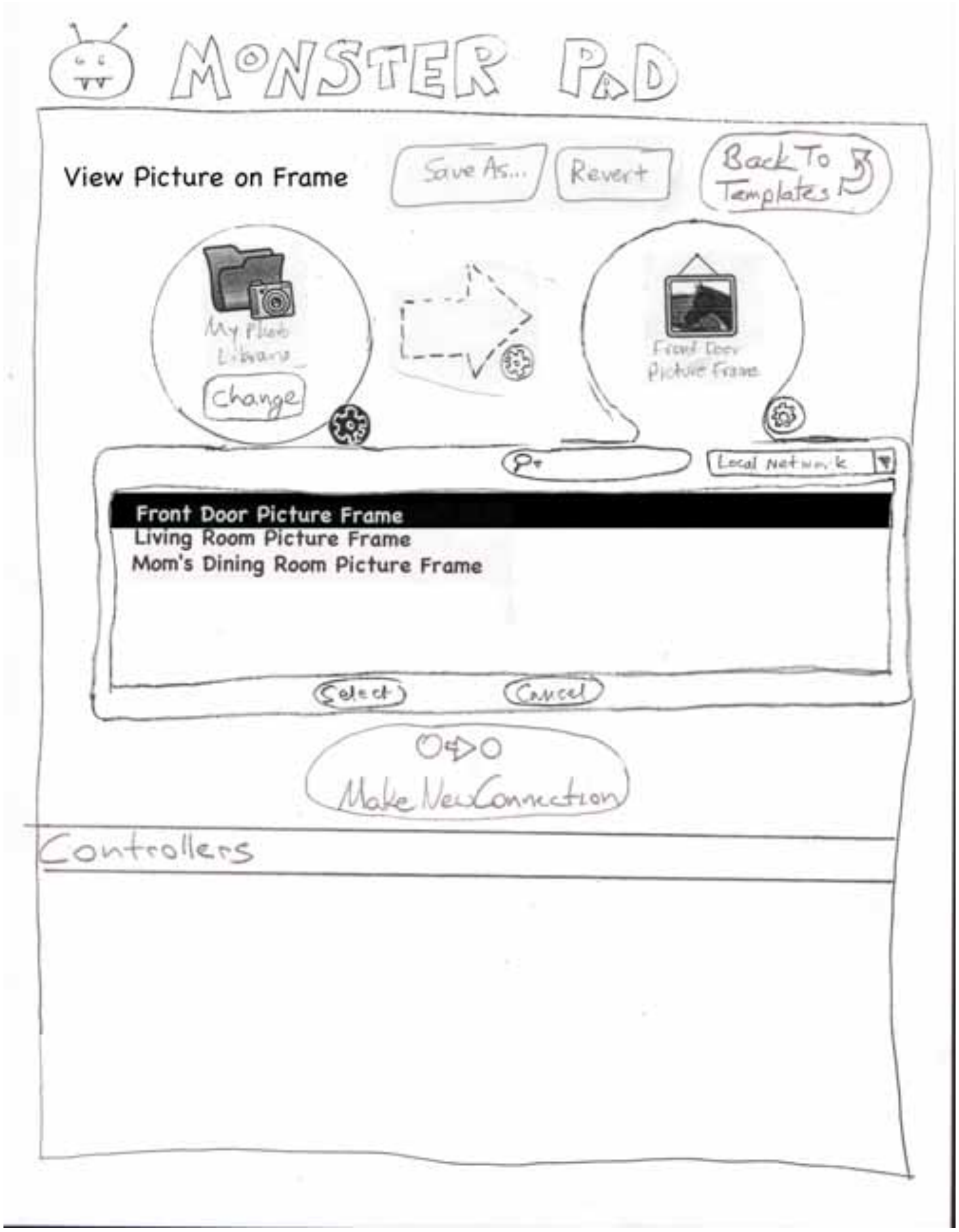
User clicks "OK"

Task3, Step 9



User clicks "Choose" under the sink slot

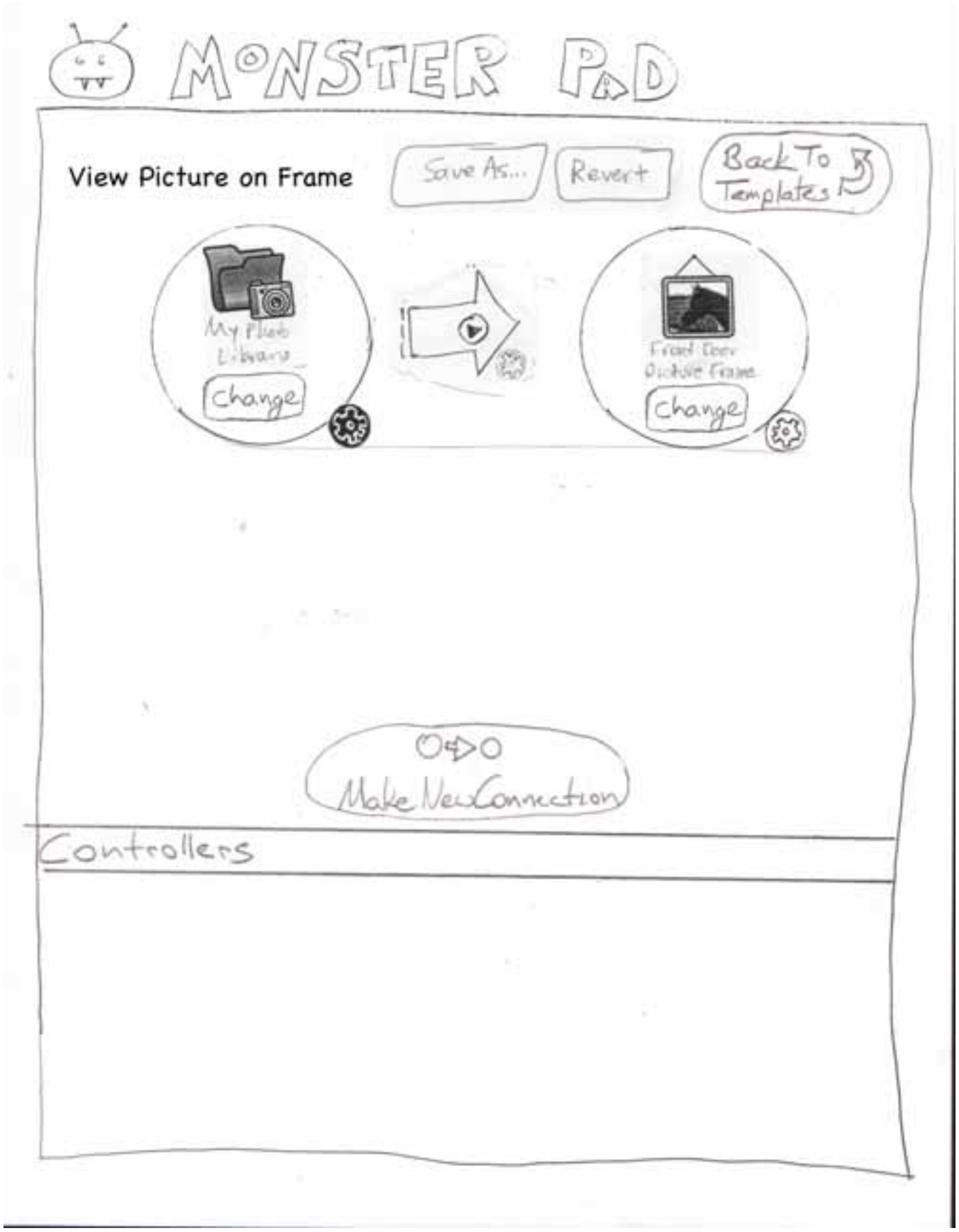
Task 3, Step 10



Front Door Picture Frame
Living Room Picture Frame
Mom's Dining Room Picture Frame

User selects "Front Door Picture Frame", then clicks "Select"

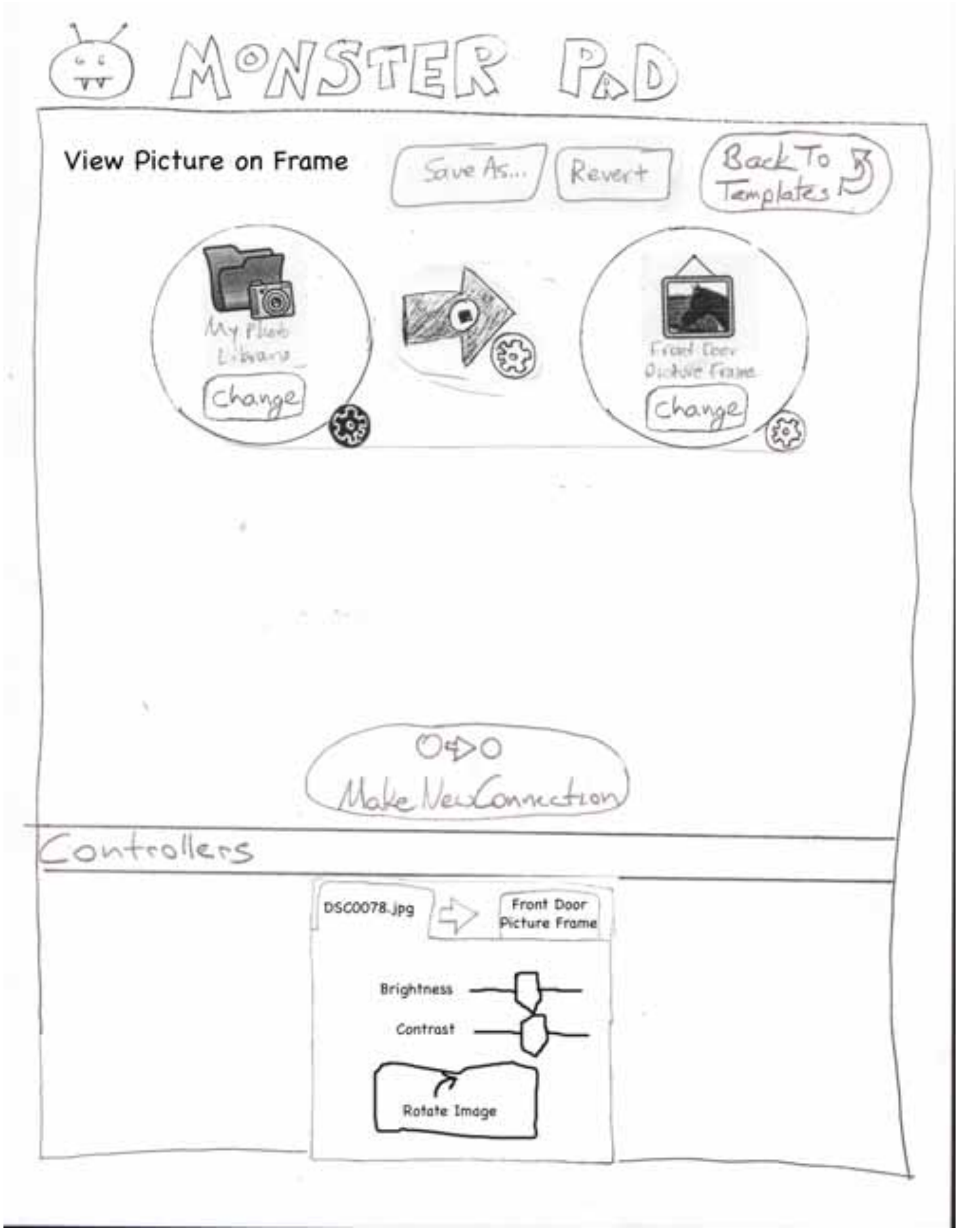
Task 3, Step 11



User clicks "Play" button

Side Effect: A random image from the photo library appears on the picture frame

Task 3, Step 12



Task is Complete

D OSCAR Expert Review #2 Materials

D.1 Instructions to Reviewers

We are primarily seeking feedback on:

- Profiles
- Walkthrough
- Recipes

…

As before, the format for your analysis and feedback is up to you. All we specifically ask is for a prioritized list of the most serious issues be included in the report.

D.1.1 Overview

The MonsterPad (name change pending) is intended to be a portable device that is used for configuring and controlling a wide variety of media and communication devices and services in the home. We imagine the MonsterPad as a Tablet-sized touch screen device with wireless connectivity. There can be multiple MonsterPads in a single home. There can also be other co-existing client devices and/or software (e.g. software that runs on a Desktop PC or on a smartphone) that performs some or all of the functions of the MonsterPad, and that can share configuration information (e.g. recipes, lists of active connections) with the MonsterPad(s).

The MonsterPad depends on the existence of a bunch of devices and services that are "Monster-enabled," which basically means that they (a) are available on the home network or Internet and (b) expose the appropriate interfaces (a.k.a. "service descriptions" or "profiles" in other interoperability frameworks like .NET and UPnP) to allow them to interoperate with other "Monster-enabled" devices and services. For the purposes of this design, we are assuming a fairly large number of Monster-enabled devices per home. We have generously placed speakers, display devices (TVs, Picture Frames), web cams, and microphones in almost every room in the house and assumed that they are all equipped with the ability to participate in the Monster network, whether because they have that capability built-in or because they have been "Monster-enabled" via some kind of proxy device.

It is not the purpose of this exercise to assess the likelihood that "Monster" or its close cousin "Obje" will achieve such remarkable (and unprecedented) market dominance, it is rather to explore the desirability of having a highly composable and configurable network of devices and services, and to determine what facilities would be necessary for a framework and for a configuration tool that supported that level of composability and configuration. Figuring out how to make a version of the "Monster" environment and MonsterPad application that meets the constraints of the current or near-term future technical environment that can be expected in realistic homes is a task for a later Phase.

D.1.2 Target Platform

The device that we are using to prototype the MonsterPad, and that we intend to use in future usability studies, is the Fujitsu Stylistic Tablet PC. This is a stylus-based pen computer, not a touch screen, but we are designing all of the interaction with the MonsterPad to work with a finger as well as a stylus as the input device.

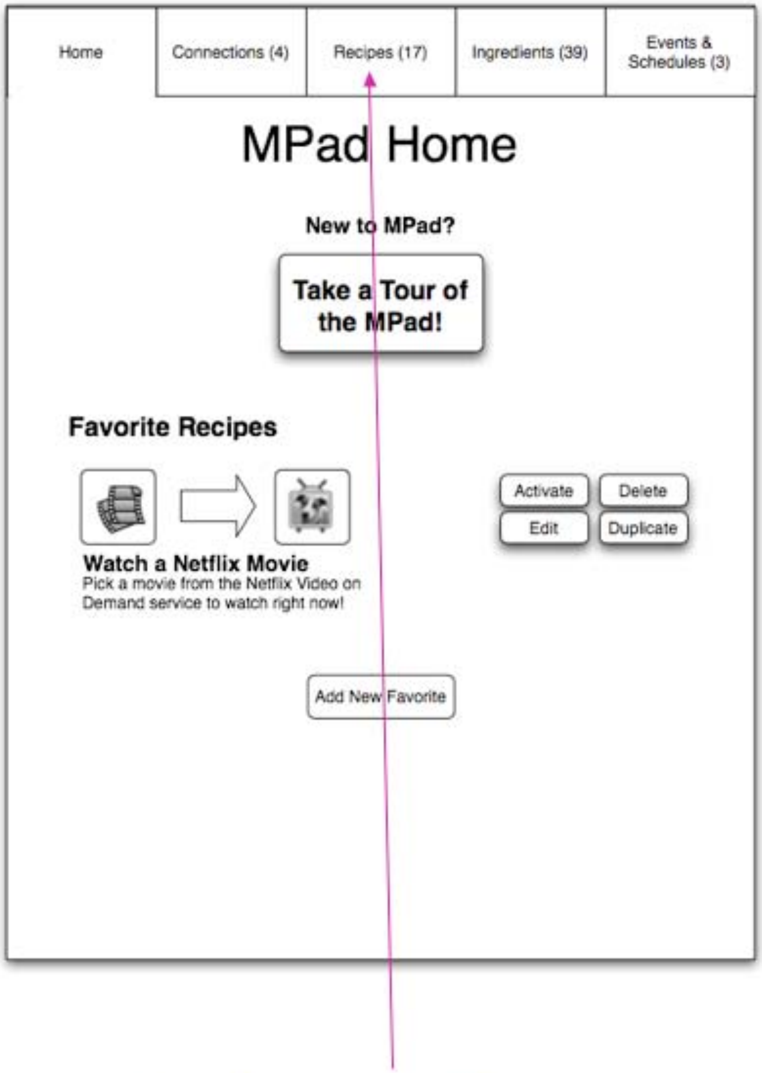
The MonsterPad application is being designed for a portrait-mode full color display, 768x1024 pixels.

D.2 Example Walkthrough: Task 3

Task 3: Show off Pictures

Dan creates a new recipe to help him show off family photos (on the living room TV) when guests come over

Task 3, Step 1



MPad Home

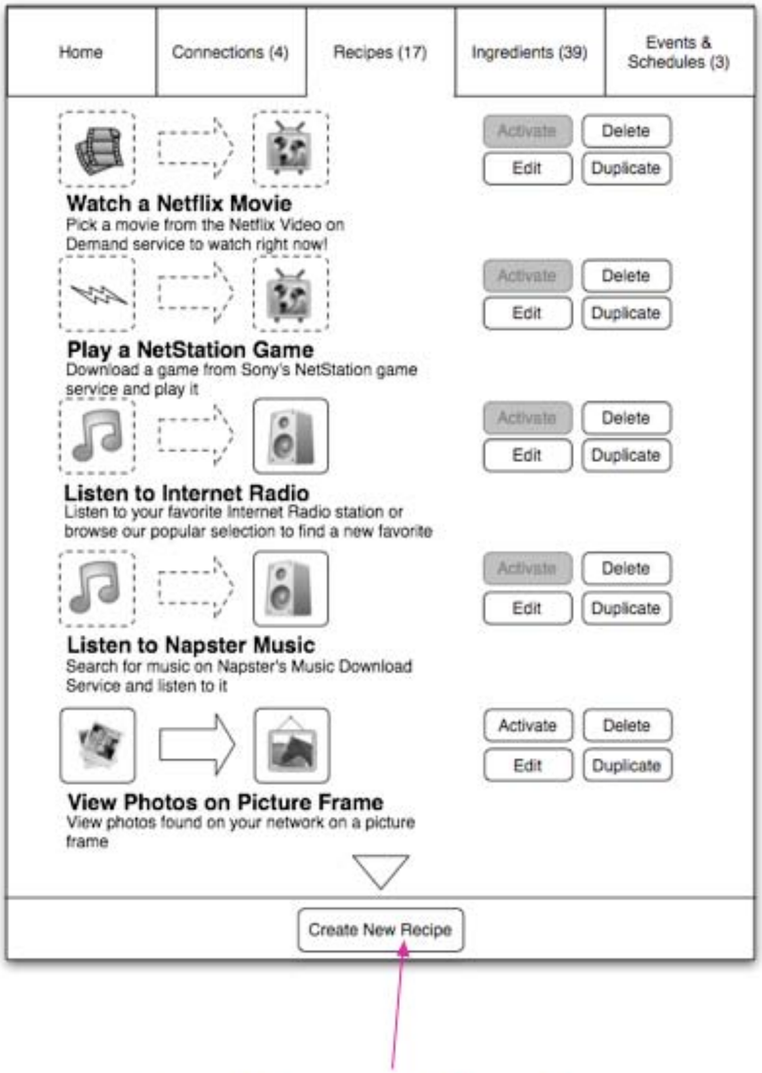
New to MPad?

Favorite Recipes

Watch a Netflix Movie
Pick a movie from the Netflix Video on Demand service to watch right now!

Touch Recipes Tab

Task 3, Step 2



Watch a Netflix Movie
Pick a movie from the Netflix Video on Demand service to watch right now!

Play a NetStation Game
Download a game from Sony's NetStation game service and play it

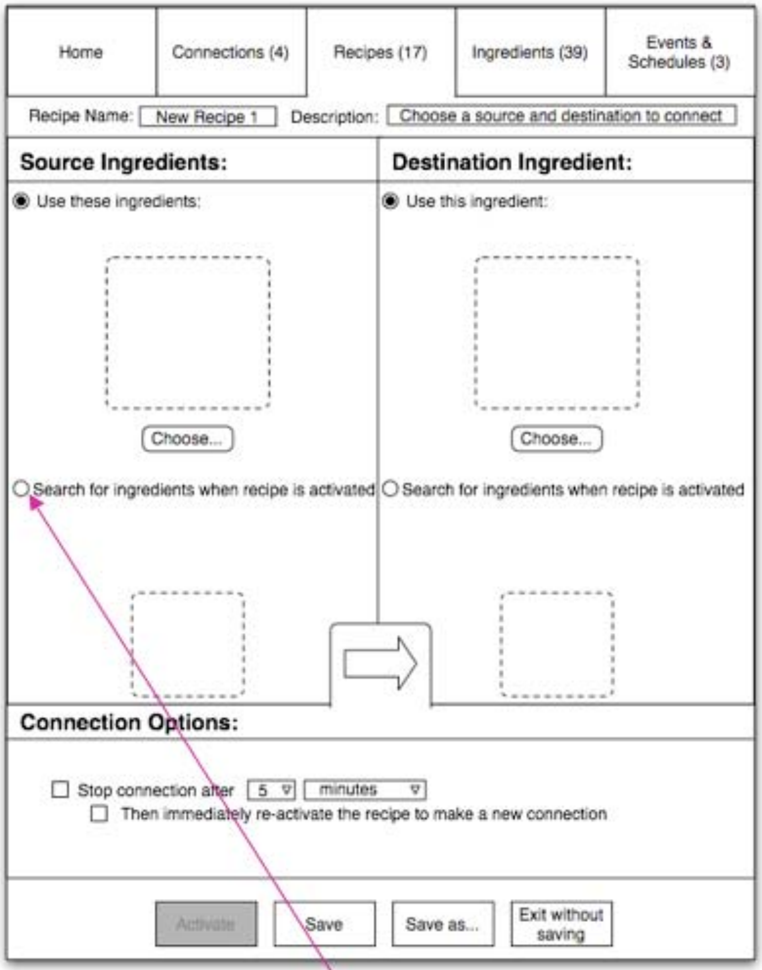
Listen to Internet Radio
Listen to your favorite Internet Radio station or browse our popular selection to find a new favorite

Listen to Napster Music
Search for music on Napster's Music Download Service and listen to it

View Photos on Picture Frame
View photos found on your network on a picture frame

Touch Create New Recipe

Task 3, Step 3



Recipe Name: New Recipe 1 Description: Choose a source and destination to connect

Source Ingredients:

● Use these ingredients:

○ Search for ingredients when recipe is activated

Destination Ingredient:

● Use this ingredient:

○ Search for ingredients when recipe is activated

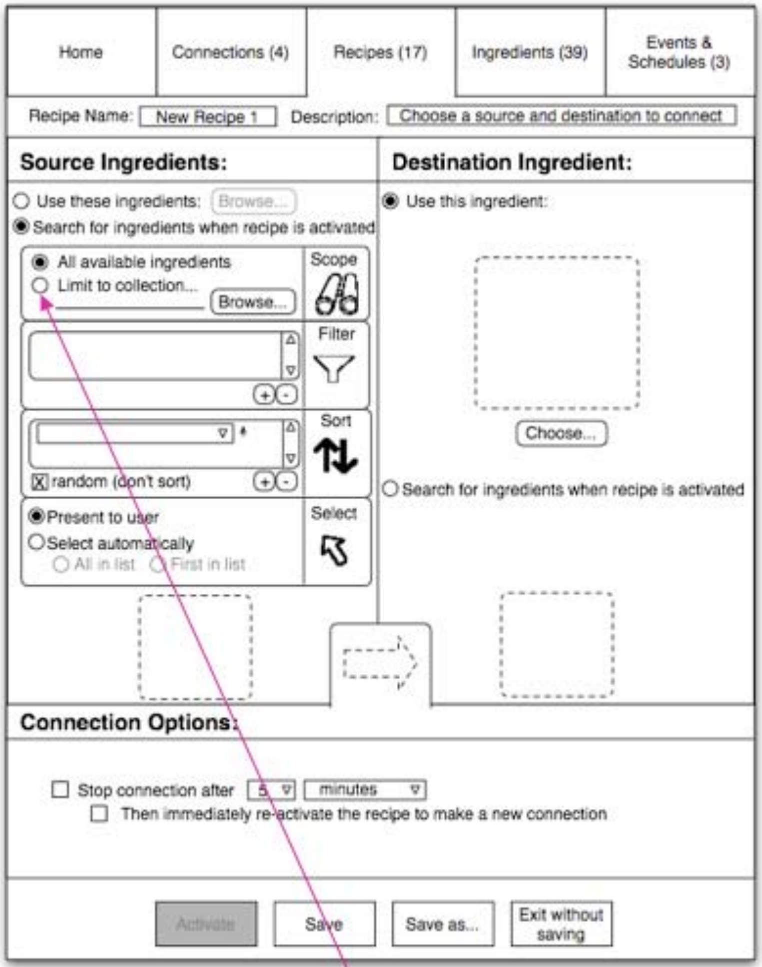
Connection Options:

□ Stop connection after 5 minutes

□ Then immediately re-activate the recipe to make a new connection

Touch “Search for indredients…”

Task 3, Step 4



Recipe Name: New Recipe 1 Description: Choose a source and destination to connect

Source Ingredients:

○ Use these ingredients: Browse...

● Search for ingredients when recipe is activated

● All available ingredients

○ Limit to collection... Browse...

☒ random (don't sort)

● Present to user

○ Select automatically

○ All in list ○ First in list

Destination Ingredient:

● Use this ingredient:

○ Search for ingredients when recipe is activated

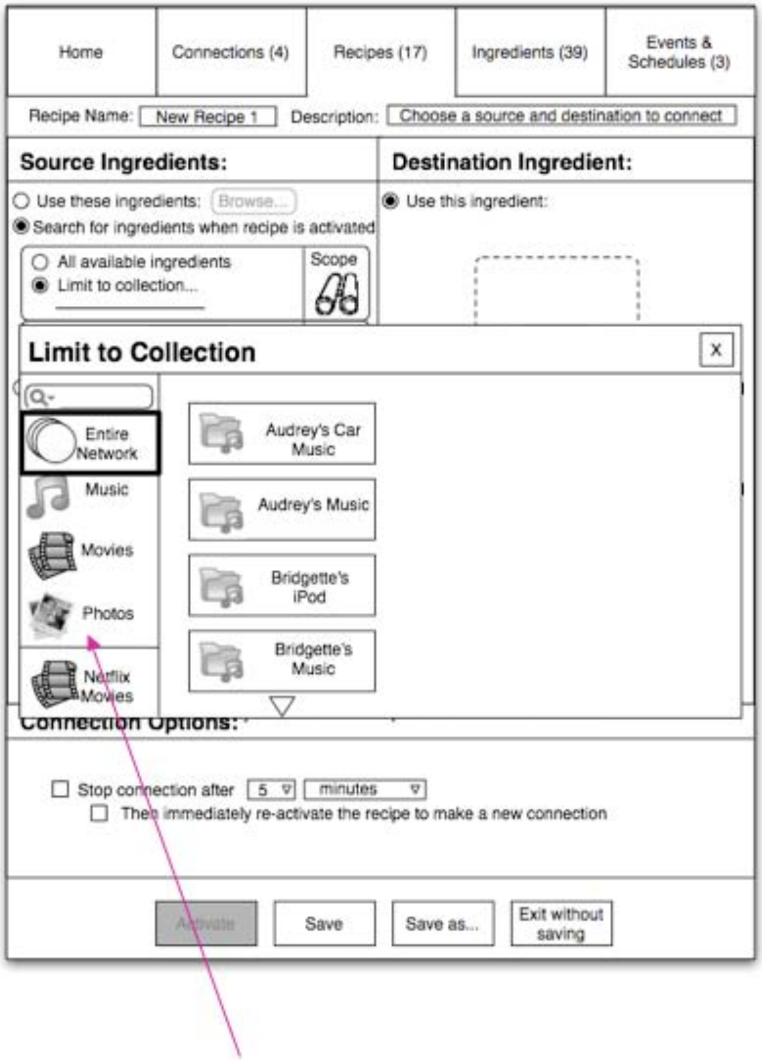
Connection Options:

□ Stop connection after 5 minutes

□ Then immediately re-activate the recipe to make a new connection

Touch Limit to Collection

Task 3, Step 5



Recipe Name: New Recipe 1 Description: Choose a source and destination to connect

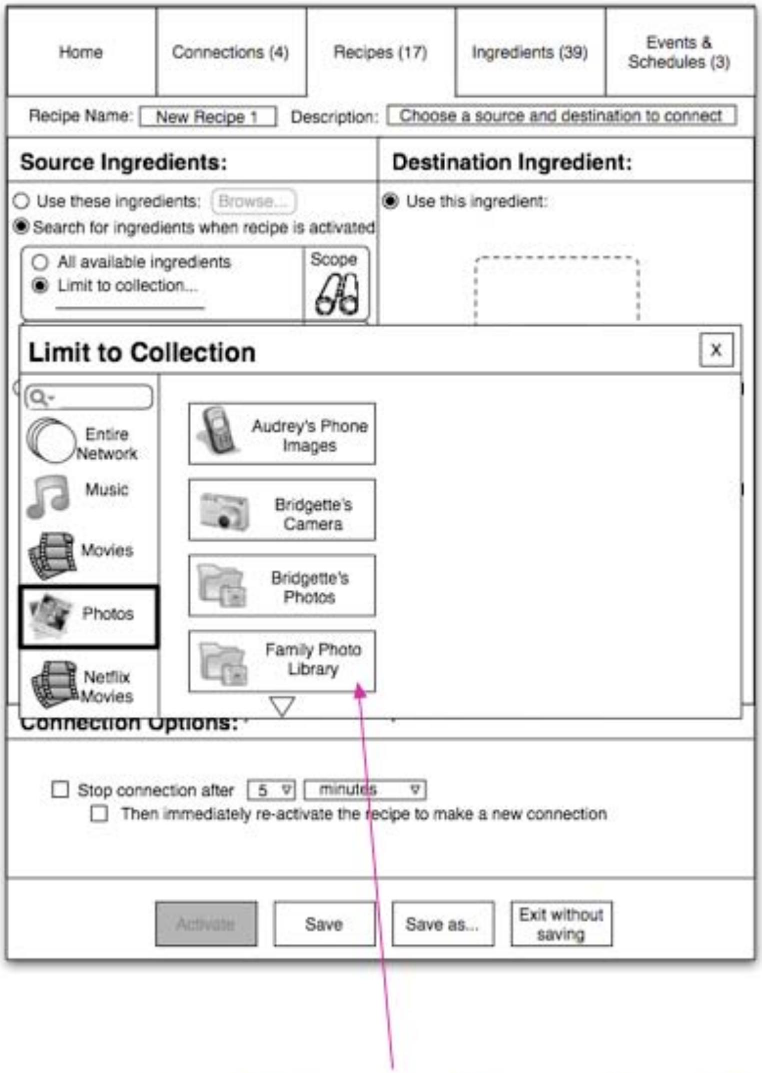
○ Use these ingredients:
● Search for ingredients when recipe is activated
○ All available ingredients
● Limit to collection...

● Use this ingredient:

□ Stop connection after 5 ▿ minutes ▿
□ Then immediately re-activate the recipe to make a new connection

Touch Photos

Task 3, Step 6



Recipe Name: New Recipe 1 Description: Choose a source and destination to connect

Source Ingredients:

○ Use these ingredients: Browse...

● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection...

Destination Ingredient:

● Use this ingredient:

Limit to Collection

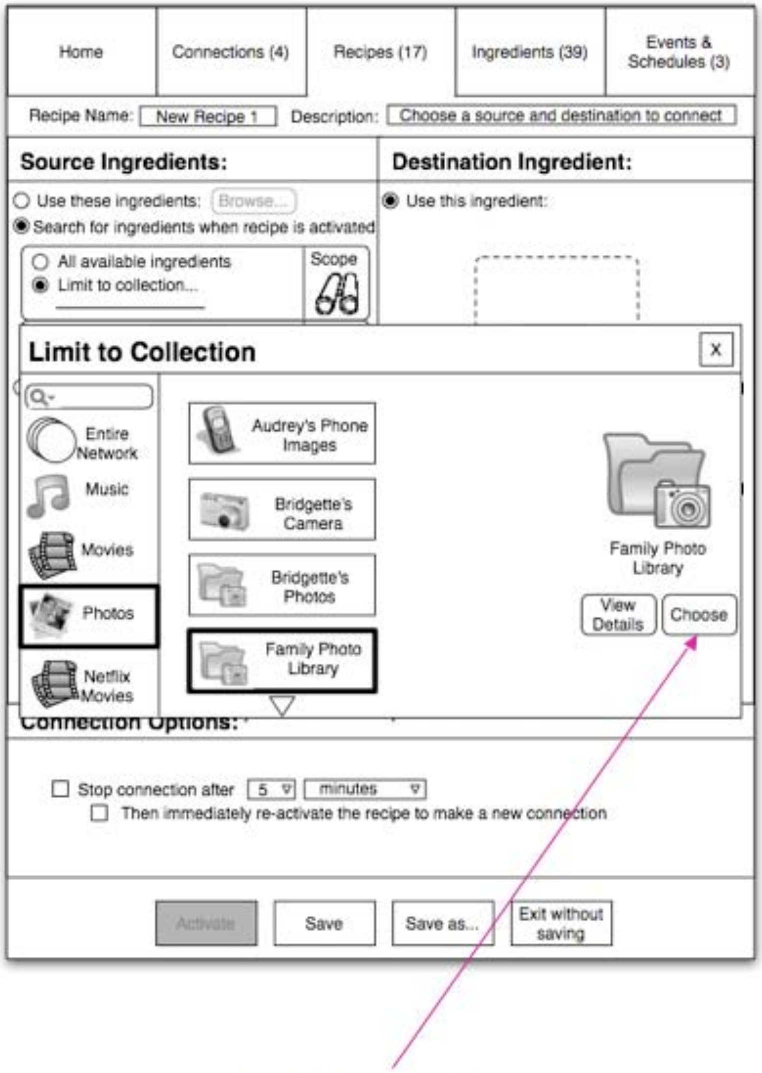
Connection Options:

□ Stop connection after 5 minutes

□ Then immediately re-activate the recipe to make a new connection

Touch “Family Photo Library”

Task 3, Step 7



Recipe Name: New Recipe 1

Description: Choose a source and destination to connect

○ Use these ingredients:

● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection...

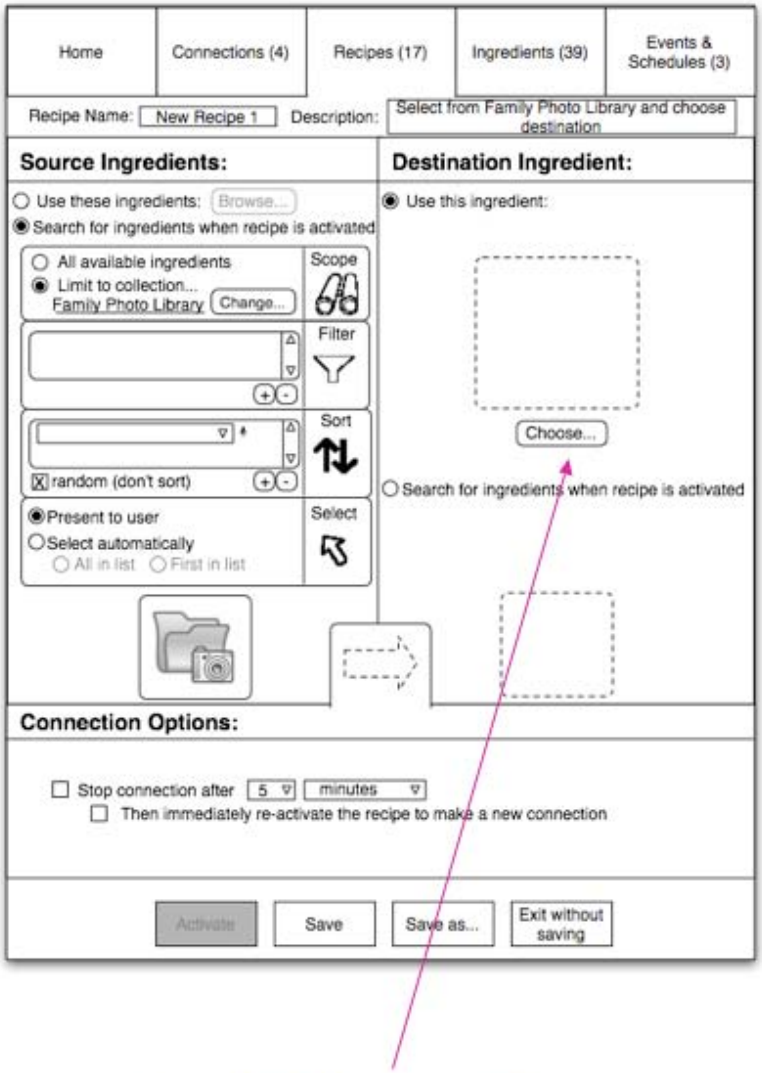
● Use this ingredient:

□ Stop connection after 5 ▿ minutes ▿

□ Then immediately re-activate the recipe to make a new connection

Touch "Choose"

Task 3, Step 8



Recipe Name: New Recipe 1 Description: Select from Family Photo Library and choose destination

Source Ingredients:

○ Use these ingredients:

● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection... Family Photo Library

☒ random (don't sort)

● Present to user

○ Select automatically

○ All in list ○ First in list

Destination Ingredient:

● Use this ingredient:

○ Search for ingredients when recipe is activated

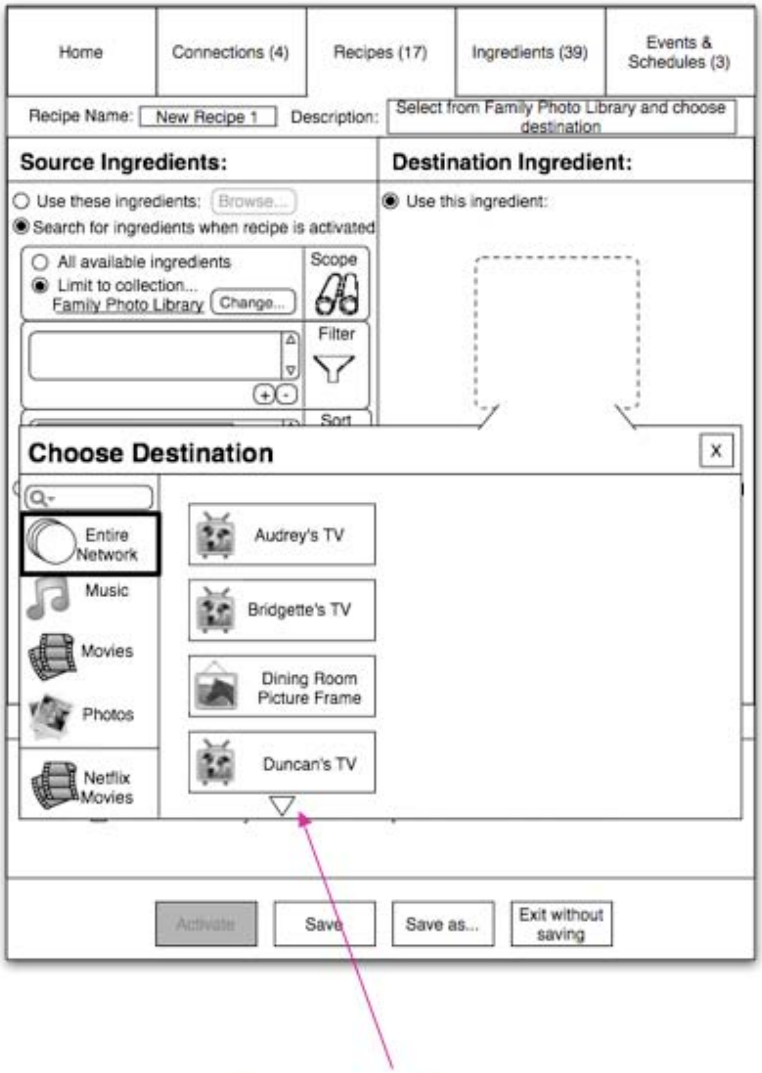
Connection Options:

□ Stop connection after 5 minutes

□ Then immediately re-activate the recipe to make a new connection

Touch "Choose..."

Task 3, Step 9



Recipe Name: New Recipe 1

Description: Select from Family Photo Library and choose destination

○ Use these ingredients:

● Search for ingredients when recipe is activated

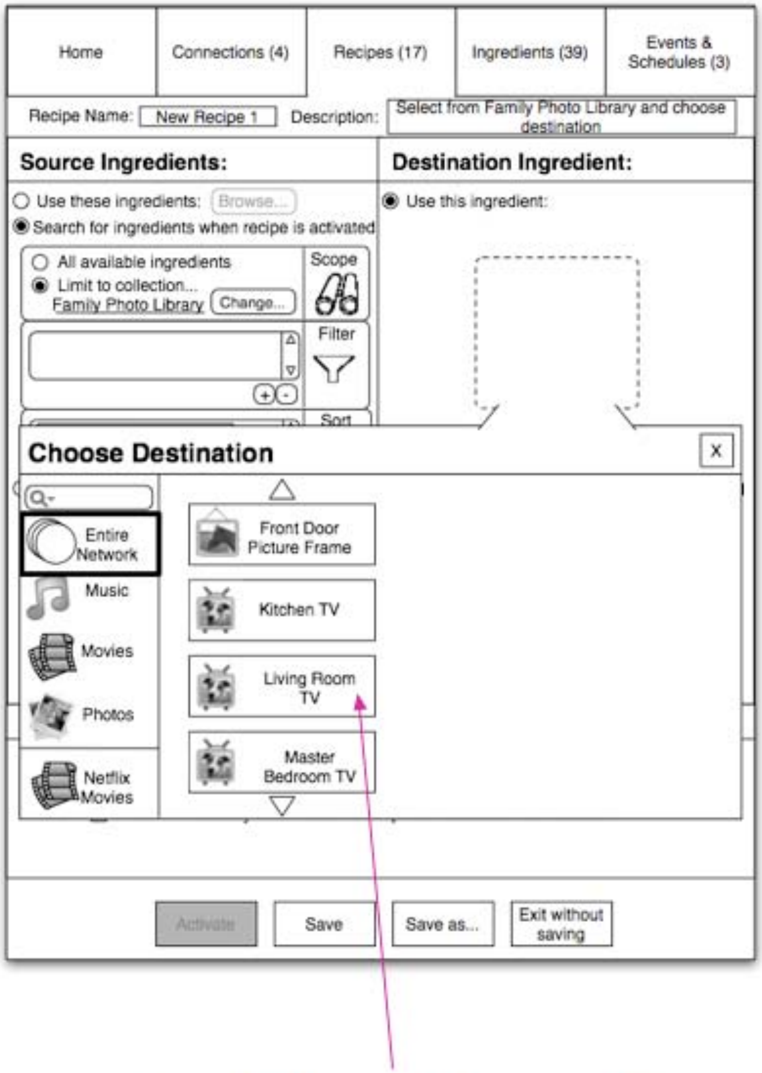
○ All available ingredients

● Limit to collection... Family Photo Library

● Use this ingredient:

Touch Down Arrow

Task 3, Step 10



Recipe Name: New Recipe 1 Description: Select from Family Photo Library and choose destination

Source Ingredients:

○ Use these ingredients: Browse...

● Search for ingredients when recipe is activated

○ All available ingredients

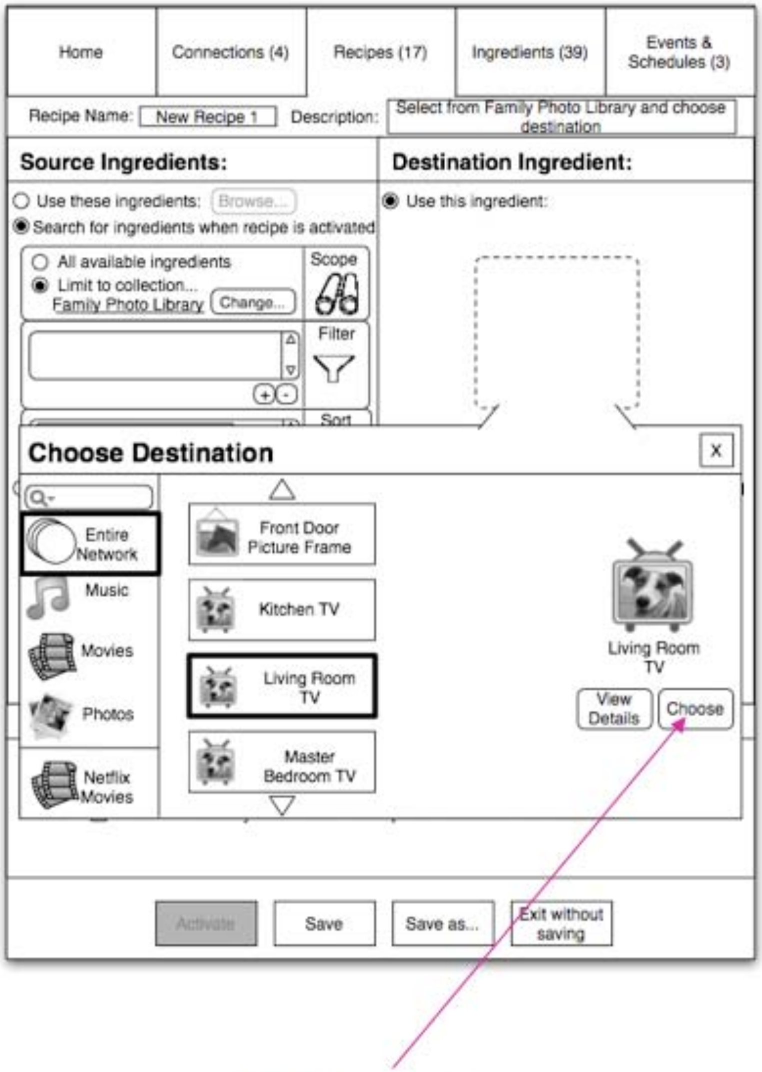
● Limit to collection... Family Photo Library Change...

Destination Ingredient:

● Use this ingredient:

Touch “Living Room TV”

Task 3, Step 11



Recipe Name: New Recipe 1 Description: Select from Family Photo Library and choose destination

Source Ingredients:

○ Use these ingredients: Browse...

● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection... Family Photo Library Change...

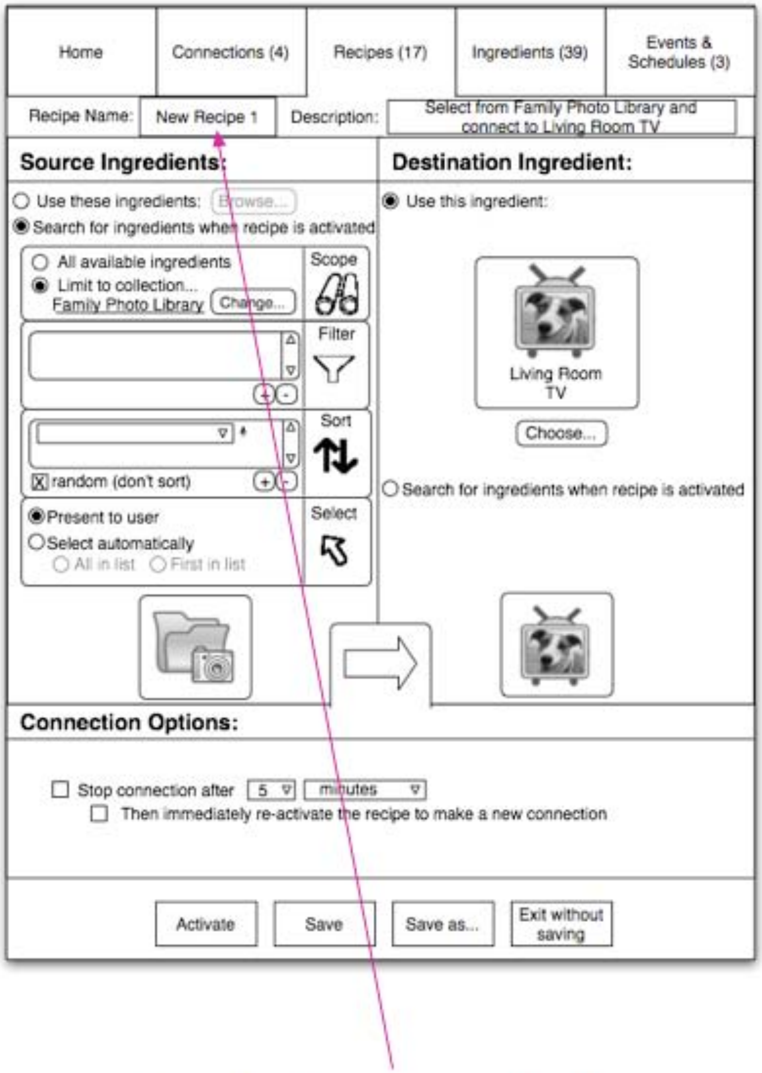
Destination Ingredient:

● Use this ingredient:

Choose Destination

Touch “Choose”

Task 3, Step 12



Recipe Name: New Recipe 1

Description: Select from Family Photo Library and connect to Living Room TV

Source Ingredients:

○ Use these ingredients:

● Search for ingredients when recipe is activated

○ All available ingredients

● Limit to collection...

Family Photo Library

☒ random (don't sort)

● Present to user

○ Select automatically

○ All in list ○ First in list

Destination Ingredient:

● Use this ingredient:

Living Room TV

○ Search for ingredients when recipe is activated

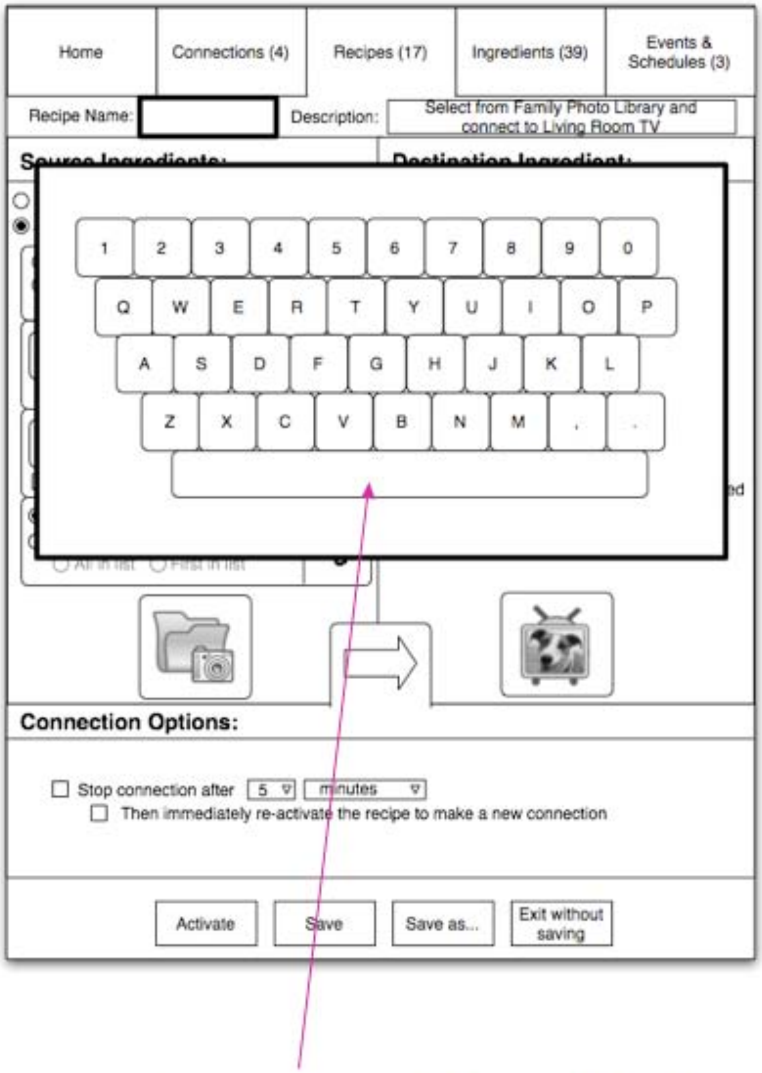
Connection Options:

□ Stop connection after 5 minutes

□ Then immediately re-activate the recipe to make a new connection

Touch Name text field

Task 3, Step 13



Recipe Name:

Description: Select from Family Photo Library and connect to Living Room TV

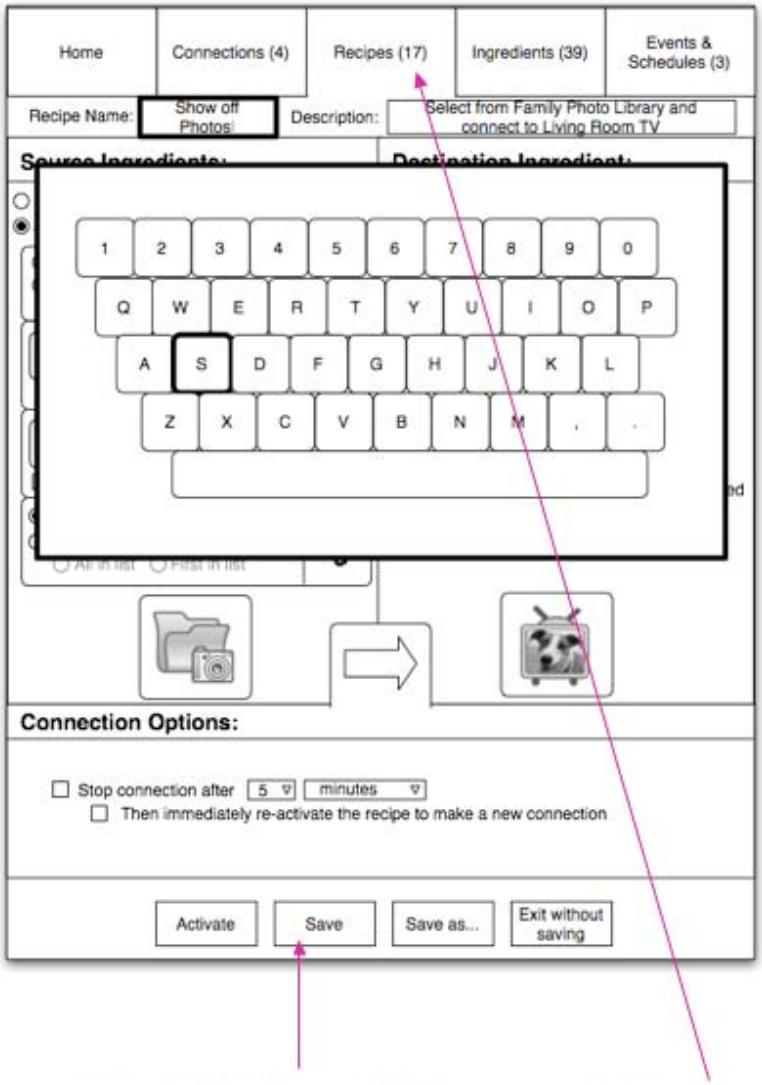
Connection Options:

☐ Stop connection after 5 ▿ minutes ▿

☐ Then immediately re-activate the recipe to make a new connection

Enter name using soft keyboard

Task 3, Step 14



Recipe Name: Show off Photos!

Description: Select from Family Photo Library and connect to Living Room TV

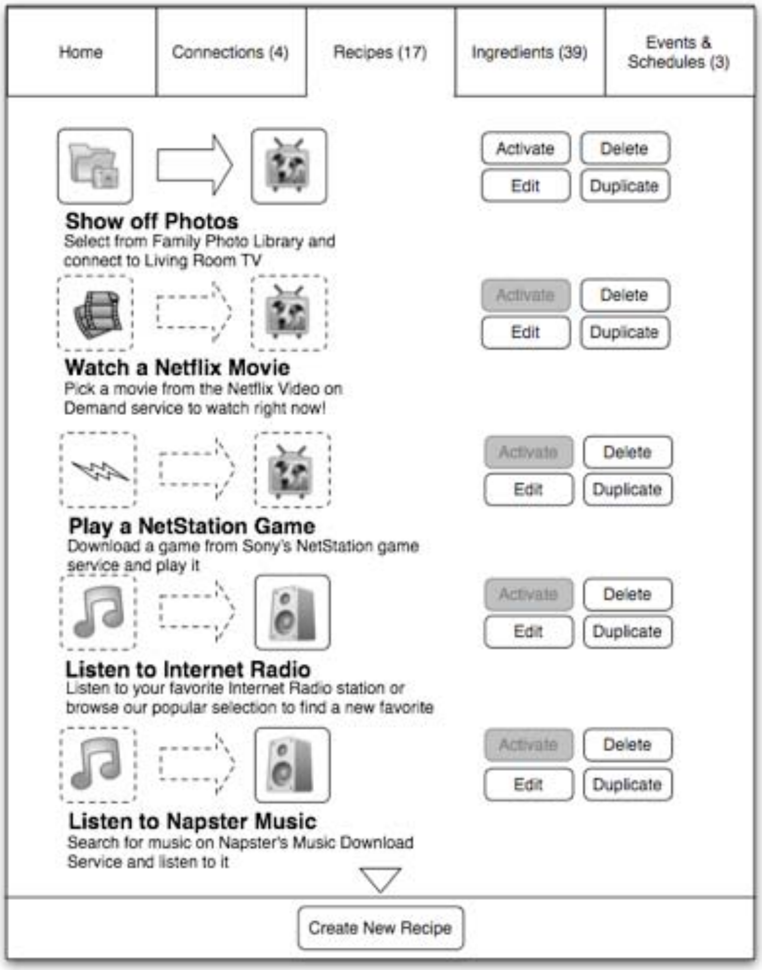
Connection Options:

☐ Stop connection after 5 ▿ minutes ▿

☐ Then immediately re-activate the recipe to make a new connection

Touch "Save", then touch "Recipes"

Task 3, Step 15



Home | Connections (4) | Recipes (17) | Ingredients (39) | Events & Schedules (3)

Show off Photos
Select from Family Photo Library and connect to Living Room TV

Activate | Delete | Edit | Duplicate

Watch a Netflix Movie
Pick a movie from the Netflix Video on Demand service to watch right now!

Activate | Delete | Edit | Duplicate

Play a NetStation Game
Download a game from Sony's NetStation game service and play it

Activate | Delete | Edit | Duplicate

Listen to Internet Radio
Listen to your favorite Internet Radio station or browse our popular selection to find a new favorite

Activate | Delete | Edit | Duplicate

Listen to Napster Music
Search for music on Napster's Music Download Service and listen to it

Activate | Delete | Edit | Duplicate

Create New Recipe

C'est tout!

E OSCAR User Study Materials

E.1 Screening questionnaire

Text to be read to respondent is in normal type.
Instructions to the interviewer are in italics.
Fields that are to be specified at the time of the interview are in <ANGLE BRACKETS>.

Number of study participants sought: 10

Eligibility requirements:

- *All participants must be 18 years of age or greater.*
- *No participants should be currently employed as programmers or system administrators*
- *No participants should be working in a high-tech industry*

We are seeking a sample that is ***as balanced as possible*** *(need not be exact, except where "0%" is specified) along the following criteria:*

- *Gender: ~50% male, ~50% female*
- *Education Level:*
    - *0% High school or less*
    - *~25% Some college*
    - *~50% College graduate OR Some post-graduate*
    - *~25% Post-graduate degree*
- *Media consumption habits*
    - *0% low*
    - *50% moderate*
    - *50% high*
- *Technology adoption habits*
    - *0% conservative or very conservative*
    - *25% late majority*
    - *50% early majority*
    - *25% early adopter*

Record responses to all questions for delivery to the research team. If respondent is deemed ineligible or declines to participate in the study, delete all records of that respondent's responses.

Hi my name is <INTERVIEWER NAME> from <RECRUITING FIRM> and we are seeking people to participate in a study regarding the usability of new software that provides new ways for consumers to interact with media devices in their homes. The study lasts 60-90 minutes and takes place at a location in Palo Alto. You will be given a choice of times in between <START DATE> and <END DATE> in which to schedule your participation. You will receive a stipend of $100 for participating, which will be delivered by check within 3 weeks after completing your participation in the study. This project is for research only and no sales or solicitation efforts will be made at any time. If you are interested in participating, I need to ask you a few questions to determine your eligibility.

If not already known
1. What is your gender?
Consult desired breakdown. If category is full, terminate interview. Not eligible.

1. What is your age?
If less than 18, terminate interview. Not eligible

2. What is your profession?
If computer programmer or system/network administrator, terminate interview. Not eligible.

3. What industry do you currently work in?
If CONSUMER ELECTRONICS, SOFTWARE, or other high-technology, terminate interview. Not eligible.

4. What is your educational background? Please choose from the following

- Some high school
- High school diploma
- Some college
- College degree
- Some post-graduate school
- Graduate or professional degree

Consult desired breakdown. If category is already full, terminate interview. Not eligible.

5. How many DVDs or videocassettes do you watch in an average month?

0-2:	*low*
3-10:	*moderate*
10+:	*high*

6. How many CDs do you purchase in an average month?

0-2: low
3-10: moderate
10+: high

7. How many hours do you spend watching TV in an average week?

0-5: low
5-15: moderate
15+: high

How to rank respondent:

2-3 Highs = high
1 High + 2 moderates = high
1 High + 1 moderate = moderate
2-3 moderates = moderate
0-1 moderates or less = low

If respondent is "low" terminate interview.
Else consult breakdown. If category is full, terminate interview. Not eligible.

8. Which of the following statements best describes you?

A. I am always the first person I know to try out new gadgets and technology.
B. I will usually try out a new gadget or technology after I've heard a couple of other people say good things about it.
C. I am willing to try out new technology but I usually wait until I know a lot of people are using it.
D. I am reluctant to try out new technology until a lot of people I know are already using it without any troubles.
E. I almost never try out new technology, even when it seems that everyone else it using it.

How to rank

A: Early adopter
B: Early majority
C: Late majority
D: Conservative
E: Very conservative

If D or E, terminate interview.
Else consult breakdown. If category is full, respondent is ineligible.

End of Interview

E.2 Interview script for OSCAR2 user study[11]

Comprehension

1. What is a setup?

2. What is a connection?

3. What are devices and media?

4. How would you describe this to someone else?

5. What is your opinion of the language used in this application, especially "Setup"?

System Feedback

6. (Give them 3 Happy Stickers, 3 Frowny Stickers & screenshot printouts) What did you particularly like about the system? What did you particularly dislike?

7. What types of activities would you use this for?

1: ____

2: ____

3: ____

[11] The version used for OSCAR1 was nearly the same, with some of the language changed (e.g. "Setup" was "Recipe" in the OSCAR1 interview). Also note that several of the user study materials, such as the questionnaire, screen-by-screen feedback screens, and recipe ranking instruments were presented in Chapter 6 and therefore not presented again here.

8. Can you think of other uses for a system like this? What else do you think it should do?

9. What type of person do you think this system is for? Would you use it personally? Who else do you know that would use this?

Setup/Activity Ranking

10. (Give them Setup ranking questionnaire)

For each ranked in the top 3, ask:

Why did you think this setup would be useful?

How do you do this kind of thing now?

How would the setup allow you to do this better?

Would someone else in your household have a different ranking?

Setup Ranking #2 (Device/Media Sharing)

11. (Give them setup ranking #2 questionnaire)

For each ranked in the top 3, ask:

Why did you think this setup would be useful?

How do you do this kind of thing now?

How would the setup allow you to do this better?

Would someone else in your household have a different ranking?

12. Would the ability to share devices and media like this make the system more useful?

Setups Sharing

13. One possible area of future development for our system is to add the ability to share setups. For example, you might give setups to your friends or family members, and they might give them to you. Also, you might be able to find setups on the Internet and download them to your house. Do you think you might find such a capability useful?

14. What role do you imagine that you would play in setup sharing? Would you create them, download them, use them?

Form Factor

15. What do you think of this tablet device for this application? Are there other form factors you think would be good?

16. Would this [OSCAR] be yours or the households? Ignoring cost, how many of these would you want to have?

Computer & Sysadmin Expertise

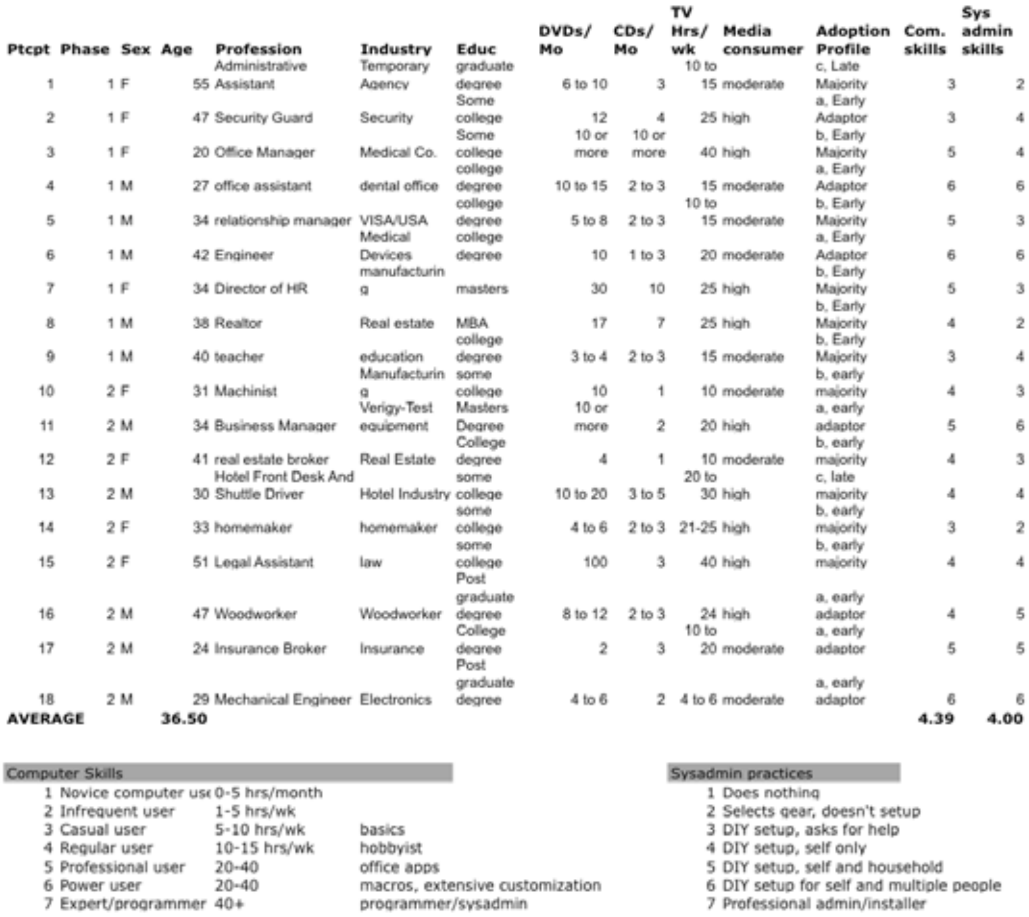
17. How would you characterize your computer expertise? How many hours per week would you say you use a computer at work? At home? What applications do you use the most frequently?

18. At home, what role do you play in selecting, installing, and maintaining computer and home a/v/ gear? Do you ever help other people with these activities?

for office use only	
COMPUTER EXPERTISE	SYADMIN RATING:
1 Novice; 0-5 hrs/month	1 Does nothing
2 Infrequent user; 1-5 hrs/wk	2 Selects gear, doesn't setup
3 Casual user; 5-10 hrs/wk	3 DIY setup, asks for help
4 Regular user; 10-15 hrs/wk	4 DIY setup, self only
5 Professional user; 20-40	5 DIY setup, self and household
6 Power user; 20-40	6 DIY setup for self and multiple people
7 Expert/programmer; 40+	7 Professional admin/installer

F OSCAR User Study Results

F.1 Demographics

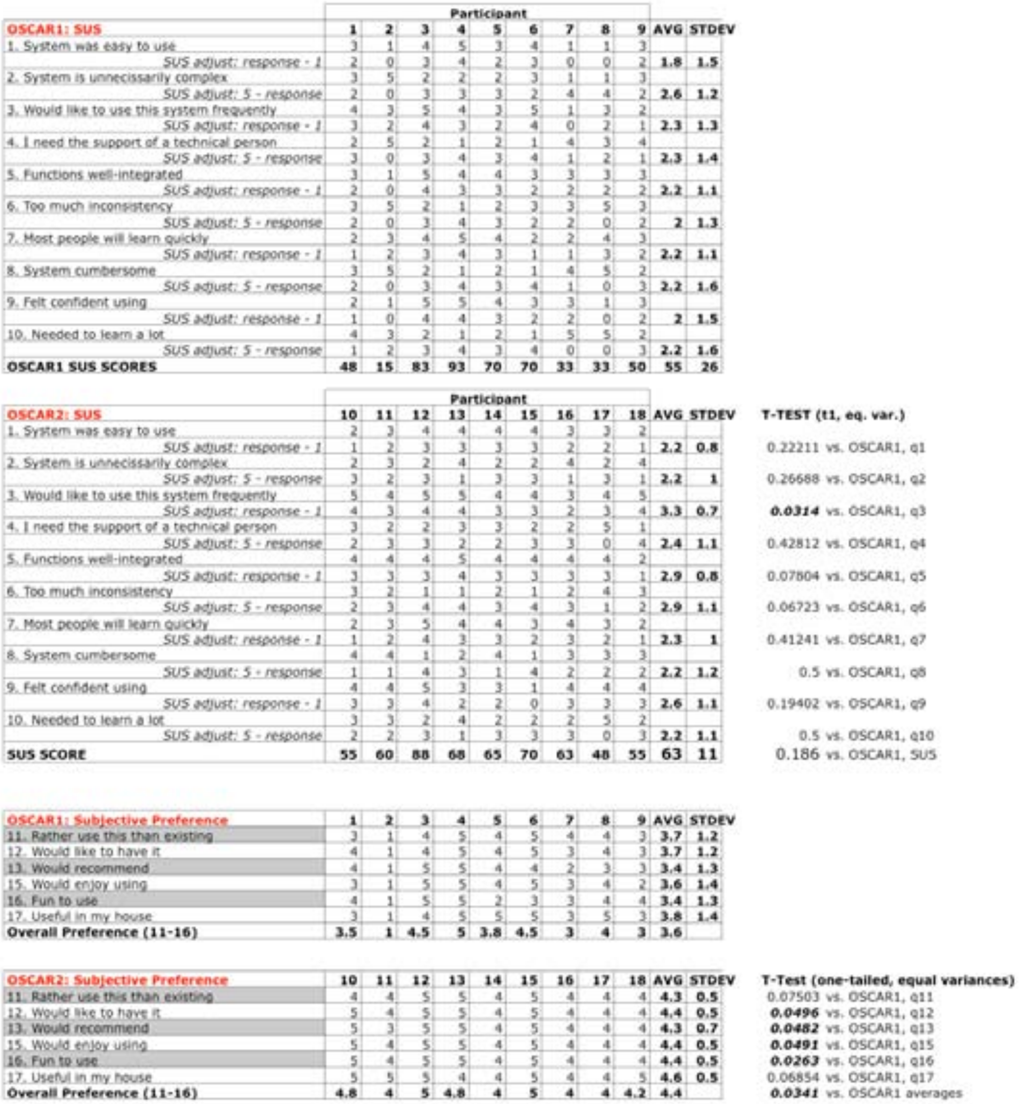


Ptcpt	Phase	Sex	Age	Profession	Industry	Educ	DVDs/ Mo	CDs/ Mo	TV Hrs/ wk	Media consumer	Adoption Profile	Com. skills	Sys admin skills
1	1	F	55	Administrative Assistant	Temporary Agency	graduate degree	6 to 10	3	10 to 15	moderate	c, Late Majority	3	2
2	1	F	47	Security Guard	Security	Some college	12	4	25	high	a, Early Adaptor	3	4
3	1	F	20	Office Manager	Medical Co.	Some college	10 or more	10 or more	40	high	b, Early Majority	5	4
4	1	M	27	office assistant	dental office	college degree	10 to 15	2 to 3	15	moderate	a, Early Adaptor	6	6
5	1	M	34	relationship manager	VISA/USA	college degree	5 to 8	2 to 3	10 to 15	moderate	b, Early Majority	5	3
6	1	M	42	Engineer	Medical Devices	college degree	10	1 to 3	20	moderate	a, Early Adaptor	6	6
7	1	F	34	Director of HR	manufacturing	masters	30	10	25	high	b, Early Majority	5	3
8	1	M	38	Realtor	Real estate	MBA	17	7	25	high	b, Early Majority	4	2
9	1	M	40	teacher	education	college degree	3 to 4	2 to 3	15	moderate	b, Early Majority	3	4
10	2	F	31	Machinist	Manufacturing	some college	10	1	10	moderate	b, early majority	4	3
11	2	M	34	Business Manager	Verigy-Test equipment	Masters Degree	10 or more	2	20	high	a, early adaptor	5	6
12	2	F	41	real estate broker	Real Estate	College degree	4	1	10	moderate	b, early majority	4	3
13	2	M	30	Hotel Front Desk And Shuttle Driver	Hotel Industry	some college	10 to 20	3 to 5	20 to 30	high	c, late majority	4	4
14	2	F	33	homemaker	homemaker	some college	4 to 6	2 to 3	21-25	high	b, early majority	3	2
15	2	F	51	Legal Assistant	law	some college	100	3	40	high	b, early majority	4	4
16	2	M	47	Woodworker	Woodworker	Post graduate degree	8 to 12	2 to 3	24	high	a, early adaptor	4	5
17	2	M	24	Insurance Broker	Insurance	College degree	2	3	10 to 20	moderate	a, early adaptor	5	5
18	2	M	29	Mechanical Engineer	Electronics	Post graduate degree	4 to 6	2	4 to 6	moderate	a, early adaptor	6	6
AVERAGE			**36.50**									**4.39**	**4.00**

Computer Skills			
1	Novice computer us	0-5 hrs/month	
2	Infrequent user	1-5 hrs/wk	
3	Casual user	5-10 hrs/wk	basics
4	Regular user	10-15 hrs/wk	hobbyist
5	Professional user	20-40	office apps
6	Power user	20-40	macros, extensive customization
7	Expert/programmer	40+	programmer/sysadmin

Sysadmin practices	
1	Does nothing
2	Selects gear, doesn't setup
3	DIY setup, asks for help
4	DIY setup, self only
5	DIY setup, self and household
6	DIY setup for self and multiple people
7	Professional admin/installer

F.2 Questionnaire responses



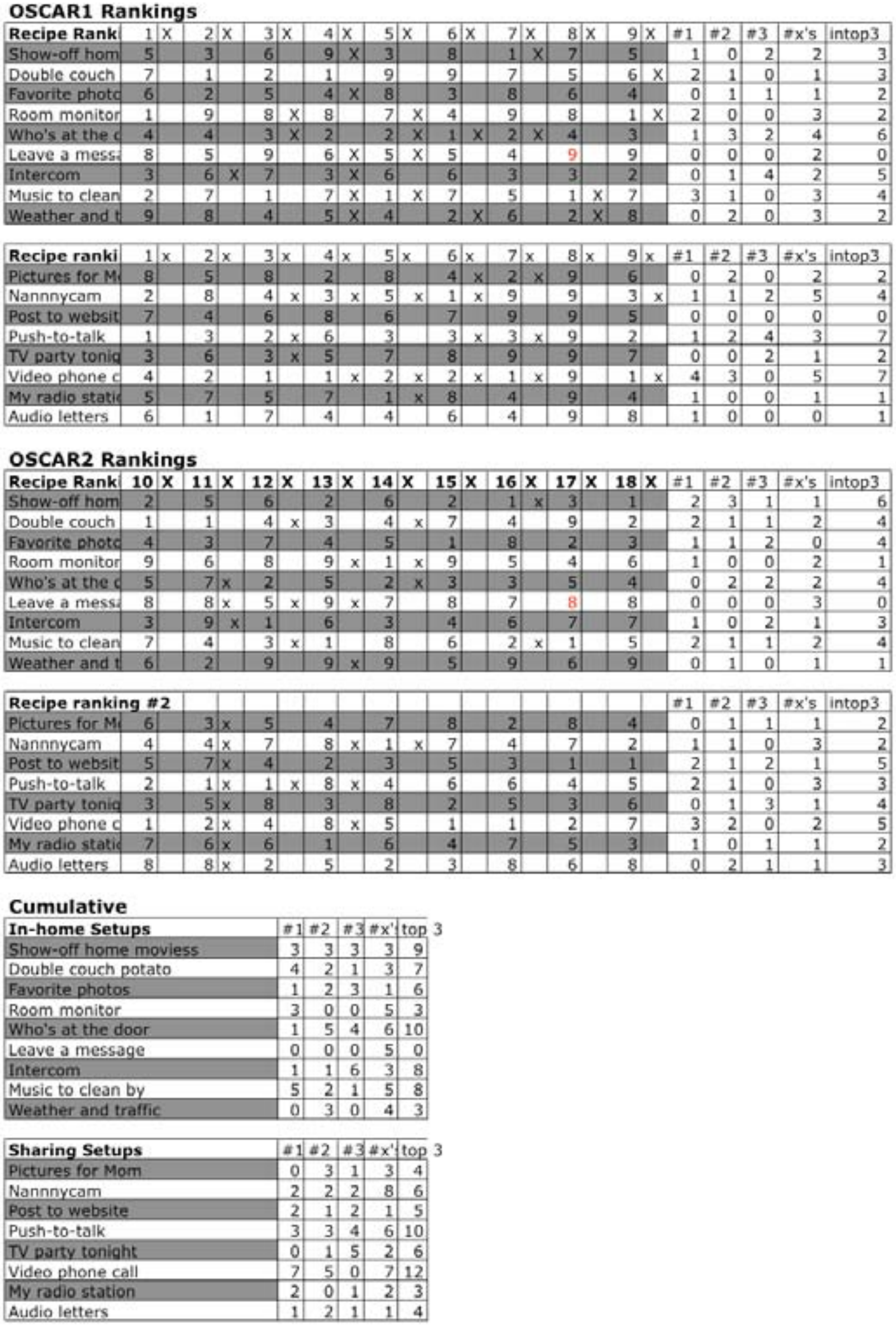
	Participant										
OSCAR1: SUS	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**AVG**	**STDEV**
1. System was easy to use	3	1	4	5	3	4	1	1	3		
SUS adjust: response - 1	2	0	3	4	2	3	0	0	2	**1.8**	**1.5**
2. System is unnecissarily complex	3	5	2	2	2	3	1	1	3		
SUS adjust: 5 - response	2	0	3	3	3	2	4	4	2	**2.6**	**1.2**
3. Would like to use this system frequently	4	3	5	4	3	5	1	3	2		
SUS adjust: response - 1	3	2	4	3	2	4	0	2	1	**2.3**	**1.3**
4. I need the support of a technical person	2	5	2	1	2	1	4	3	4		
SUS adjust: 5 - response	3	0	3	4	3	4	1	2	1	**2.3**	**1.4**
5. Functions well-integrated	3	1	5	4	4	3	3	3	3		
SUS adjust: response - 1	2	0	4	3	3	2	2	2	2	**2.2**	**1.1**
6. Too much inconsistency	3	5	2	1	2	3	3	5	3		
SUS adjust: 5 - response	2	0	3	4	3	2	2	0	2	**2**	**1.3**
7. Most people will learn quickly	2	3	4	5	4	2	2	4	3		
SUS adjust: response - 1	1	2	3	4	3	1	1	3	2	**2.2**	**1.1**
8. System cumbersome	3	5	2	1	2	1	4	5	2		
SUS adjust: 5 - response	2	0	3	4	3	4	1	0	3	**2.2**	**1.6**
9. Felt confident using	2	1	5	5	4	3	3	1	3		
SUS adjust: response - 1	1	0	4	4	3	2	2	0	2	**2**	**1.5**
10. Needed to learn a lot	4	3	2	1	2	1	5	5	2		
SUS adjust: 5 - response	1	2	3	4	3	4	0	0	3	**2.2**	**1.6**
OSCAR1 SUS SCORES	**48**	**15**	**83**	**93**	**70**	**70**	**33**	**33**	**50**	**55**	**26**

	Participant											
OSCAR2: SUS	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**AVG**	**STDEV**	**T-TEST (t1, eq. var.)**
1. System was easy to use	2	3	4	4	4	4	3	3	2			
SUS adjust: response - 1	1	2	3	3	3	3	2	2	1	**2.2**	**0.8**	0.22211 vs. OSCAR1, q1
2. System is unnecissarily complex	2	3	2	4	2	2	4	2	4			
SUS adjust: 5 - response	3	2	3	1	3	3	1	3	1	**2.2**	**1**	0.26688 vs. OSCAR1, q2
3. Would like to use this system frequently	5	4	5	5	4	4	3	4	5			
SUS adjust: response - 1	4	3	4	4	3	3	2	3	4	**3.3**	**0.7**	***0.0314*** vs. OSCAR1, q3
4. I need the support of a technical person	3	2	2	3	3	2	2	5	1			
SUS adjust: 5 - response	2	3	3	2	2	3	3	0	4	**2.4**	**1.1**	0.42812 vs. OSCAR1, q4
5. Functions well-integrated	4	4	4	5	4	4	4	4	2			
SUS adjust: response - 1	3	3	3	4	3	3	3	3	1	**2.9**	**0.8**	0.07804 vs. OSCAR1, q5
6. Too much inconsistency	3	2	1	1	2	1	2	4	3			
SUS adjust: 5 - response	2	3	4	4	3	4	3	1	2	**2.9**	**1.1**	0.06723 vs. OSCAR1, q6
7. Most people will learn quickly	2	3	5	4	4	3	4	3	2			
SUS adjust: response - 1	1	2	4	3	3	2	3	2	1	**2.3**	**1**	0.41241 vs. OSCAR1, q7
8. System cumbersome	4	4	1	2	4	1	3	3	3			
SUS adjust: 5 - response	1	1	4	3	1	4	2	2	2	**2.2**	**1.2**	0.5 vs. OSCAR1, q8
9. Felt confident using	4	4	5	3	3	1	4	4	4			
SUS adjust: response - 1	3	3	4	2	2	0	3	3	3	**2.6**	**1.1**	0.19402 vs. OSCAR1, q9
10. Needed to learn a lot	3	3	2	4	2	2	2	5	2			
SUS adjust: 5 - response	2	2	3	1	3	3	3	0	3	**2.2**	**1.1**	0.5 vs. OSCAR1, q10
SUS SCORE	**55**	**60**	**88**	**68**	**65**	**70**	**63**	**48**	**55**	**63**	**11**	0.186 vs. OSCAR1, SUS

OSCAR1: Subjective Preference	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**AVG**	**STDEV**
11. Rather use this than existing	3	1	4	5	4	5	4	4	3	**3.7**	**1.2**
12. Would like to have it	4	1	4	5	4	5	3	4	3	**3.7**	**1.2**
13. Would recommend	4	1	5	5	4	4	2	3	3	**3.4**	**1.3**
15. Would enjoy using	3	1	5	5	4	5	3	4	2	**3.6**	**1.4**
16. Fun to use	4	1	5	5	2	3	3	4	4	**3.4**	**1.3**
17. Useful in my house	3	1	4	5	5	5	3	5	3	**3.8**	**1.4**
Overall Preference (11-16)	**3.5**	**1**	**4.5**	**5**	**3.8**	**4.5**	**3**	**4**	**3**	**3.6**	

OSCAR2: Subjective Preference	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**AVG**	**STDEV**	**T-Test (one-tailed, equal variances)**
11. Rather use this than existing	4	4	5	5	4	5	4	4	4	**4.3**	**0.5**	0.07503 vs. OSCAR1, q11
12. Would like to have it	5	4	5	5	4	5	4	4	4	**4.4**	**0.5**	***0.0496*** vs. OSCAR1, q12
13. Would recommend	5	3	5	5	4	5	4	4	4	**4.3**	**0.7**	***0.0482*** vs. OSCAR1, q13
15. Would enjoy using	5	4	5	5	4	5	4	4	4	**4.4**	**0.5**	***0.0491*** vs. OSCAR1, q15
16. Fun to use	5	4	5	5	4	5	4	4	4	**4.4**	**0.5**	***0.0263*** vs. OSCAR1, q16
17. Useful in my house	5	5	5	4	4	5	4	4	5	**4.6**	**0.5**	0.06854 vs. OSCAR1, q17
Overall Preference (11-16)	**4.8**	**4**	**5**	**4.8**	**4**	**5**	**4**	**4**	**4.2**	**4.4**		***0.0341*** vs. OSCAR1 averages

F.3 Recipe/Setup ranking results



OSCAR1 Rankings

Recipe Rank	1	X	2	X	3	X	4	X	5	X	6	X	7	X	8	X	9	X	#1	#2	#3	#x's	intop3
Show-off hom	5		3		6		9	X	3		8		1	X	7		5		1	0	2	2	3
Double couch	7		1		2		1		9		9		7		5		6	X	2	1	0	1	3
Favorite phot	6		2		5		4	X	8		3		8		6		4		0	1	1	1	2
Room monitor	1		9		8	X	8		7	X	4		9		8		1	X	2	0	0	3	2
Who's at the	4		4		3	X	2		2	X	1	X	2	X	4		3		1	3	2	4	6
Leave a mess	8		5		9		6	X	5	X	5		4		9		9		0	0	0	2	0
Intercom	3		6	X	7		3	X	6		6		3		3		2		0	1	4	2	5
Music to clean	2		7		1		7	X	1	X	7		5		1	X	7		3	1	0	3	4
Weather and t	9		8		4		5	X	4		2	X	6		2	X	8		0	2	0	3	2

Recipe ranki	1	x	2	x	3	x	4	x	5	x	6	x	7	x	8	x	9	x	#1	#2	#3	#x's	intop3
Pictures for M	8		5		8		2		8		4	x	2	x	9		6		0	2	0	2	2
Nannnycam	2		8		4	x	3	x	5	x	1	x	9		9		3	x	1	1	2	5	4
Post to websit	7		4		6		8		6		7		9		9		5		0	0	0	0	0
Push-to-talk	1		3		2	x	6		3		3	x	3	x	9		2		1	2	4	3	7
TV party toni	3		6		3	x	5		7		8		9		9		7		0	0	2	1	2
Video phone c	4		2		1		1	x	2	x	2	x	1	x	9		1	x	4	3	0	5	7
My radio stati	5		7		5		7		1	x	8		4		9		4		1	0	0	1	1
Audio letters	6		1		7		4		4		6		4		9		8		1	0	0	0	1

OSCAR2 Rankings

Recipe Rank	10	X	11	X	12	X	13	X	14	X	15	X	16	X	17	X	18	X	#1	#2	#3	#x's	intop3
Show-off hom	2		5		6		2		6		2		1	x	3		1		2	3	1	1	6
Double couch	1		1		4	x	3		4	x	7		4		9		2		2	1	1	2	4
Favorite phot	4		3		7		4		5		1		8		2		3		1	1	2	0	4
Room monitor	9		6		8		9	x	1	x	9		5		4		6		1	0	0	2	1
Who's at the	5		7	x	2		5		2	x	3		3		5		4		0	2	2	2	4
Leave a mess	8		8	x	5	x	9	x	7		8		7		8		8		0	0	0	3	0
Intercom	3		9	x	1		6		3		4		6		7		7		1	0	2	1	3
Music to clean	7		4		3	x	1		8		6		2	x	1		5		2	1	1	2	4
Weather and t	6		2		9		9	x	9		5		9		6		9		0	1	0	1	1

Recipe ranking #2																			#1	#2	#3	#x's	intop3
Pictures for M	6		3	x	5		4		7		8		2		8		4		0	1	1	1	2
Nannnycam	4		4	x	7		8	x	1	x	7		4		7		2		1	1	0	3	2
Post to websit	5		7	x	4		2		3		5		3		1		1		2	1	2	1	5
Push-to-talk	2		1	x	1	x	8	x	4		6		6		4		5		2	1	0	3	3
TV party toni	3		5	x	8		3		8		2		5		3		6		0	1	3	1	4
Video phone c	1		2	x	4		8	x	5		1		1		2		7		3	2	0	2	5
My radio stati	7		6	x	6		1		6		4		7		5		3		1	0	1	1	2
Audio letters	8		8	x	2		5		2		3		8		6		8		0	2	1	1	3

Cumulative

In-home Setups	#1	#2	#3	#x'	top 3
Show-off home moviess	3	3	3	3	9
Double couch potato	4	2	1	3	7
Favorite photos	1	2	3	1	6
Room monitor	3	0	0	5	3
Who's at the door	1	5	4	6	10
Leave a message	0	0	0	5	0
Intercom	1	1	6	3	8
Music to clean by	5	2	1	5	8
Weather and traffic	0	3	0	4	3

Sharing Setups	#1	#2	#3	#x'	top 3
Pictures for Mom	0	3	1	3	4
Nannnycam	2	2	2	8	6
Post to website	2	1	2	1	5
Push-to-talk	3	3	4	6	10
TV party tonight	0	1	5	2	6
Video phone call	7	5	0	7	12
My radio station	2	0	1	2	3
Audio letters	1	2	1	1	4

F.4 Interview response notes

How would you describe this to someone else?

1 touch screen, little pictures to other things

2 you can view your photos, play music, and watch TV

All–in-one remote, but better because its more advanced like
3 a computer.

way to have devices interact with eachother in a simple and
4 easy to use way

5 Enables wireless functionality ...

All in one media center. Everthing purchased as individual
6 comonments, miles of connectors coming in

Using multimedia consumer electronics devices, multifunction
remote. It’s a bigger version of a Treo (has one). But Treo
would make sense for this, safer to stay in house, don't loose
7 it.

It's neat but should be better.... You're able to sit on couch
8 and do everything.

gadget that allows things that are technical to be used to
simplify one's home, to... gather one's home entertainment,
9 ... to manipulated it in one console

Home theater universal remote control
Like back to the future movie
Something people expected we'd have by now
10 Intended to simplify electronic devices in the home

everything at the touch of a button
11 integrates all the devices in the home (does a good job)

remote control on steroids
being able to control different rooms at once
12 like a "menu"--a host of choices

13 remote control for the whole house

newer technology, initially intimidating
better than the remote control (we have 5 TVs)
don't have to lift a finger
14 lazy

15 way to integrate all your av assets

16 home automated system

17 remote control

interactive home theater
18 home control

What types of activities would you use something like this for? (open-ended version)

she likes surveillance applications (keeping track of mom, see who's at the door). Lives with mom who has dementia--Mom gets into trouble in the kitchen. Kitchen video camera would be good.
Don't want a lot of TVs in the house (don't keep TVs in bedroom).
Music, currently brings radio from room to room ("music is
1 nice").

would use music, photos, DVDs. Web cam too. Other uses could be telephone caller ID, see if it’s a solictor, TV programs, just punch in your favorites.
owns an HP iPhoto (probably "Photo Smart") device for showing pictures on TV
thinking about getting one of "those Bose things"
2 likes it as an "all-in-one" remote, no clutter

TV, music, security (who are the dogs barking at). good for a party, control what's going on in different rooms. Its hard to find all your remotes. TV remote, comcast remote, ipod
3 remote, other radio remote, TV in kitchen remote.

Has a Nintendo DS, this is familiar
Would set up a slide show on a screen or TV for trip to Hawaii
See who's at the door is "very cool", Mom would like that a lot, too.
4 Like to "connect to all parts of the house."

Music is crucial. Would use to access PC via TV, to do email in
5 a different room.

Share cable tuner across multiple rooms (later says that wife won't let him watch TV at home)
Check front door while away (mentioned this before sharing was introduced)
Has an X10 camera at the front door and some kind of wacky 2.5GHz ham radio video apparatus for viewing the image
Have music automatically follow me from room to room.
Digital photos on laptop and carry it around now, DVD, but
6 display on DVD or laptop when friends come.

7	when describing activities, lists the devices she already has, including karaoke machine and DVD-R other uses = security and car, but... "too much integration is scary" (what happens when it all malfunctions?) "Almost like taking over the universe." potential for evil in normal home use. "I do believe in hidden cameras" insinuation that too much technology is bad and makes you lazy "I worry a lot about security" (I'm not sure if this is a reference to cameras or to viruses/hackers)
8	"I'd buy one today if it had the option" to play music in multiple rooms simultaneously. (Music includes streaming radio stations.) Video from the Internet VOIP anywhere in the house. Online Radio, Local Radio & TV stations Might encourage kids to be lazy ("get up and see who's at the door")
9	Has to be really quick; everybody wants everything really quick slight concern that this type of thing encourages laziness Novel ideas--parent-teacher conference, long distance call, long distance who's at the door DVD , transfer to the car. Go to the computer to see security camera, see if you should let them in (look up person in internet databaase).
10	1. laptop for internet, music, & movies (DVD) (connect with other devices) 2. switch from tv to door camera (see who's at the door from the TV) 3. Use with downloaded music
11	1. Parties, would make it more social, have different things going on in different areas 2. multitasking: take stuff off iPod while other stuff going on 3. Picture in picture: TV & movie at the same time 4. Favorite radio stations at the push of a button
12	1. music in different rooms 2. security cameras on front and back doors 3. for real estate agents: staging and virtual tours--info about different rooms, different parts of the house 4. security, remote house sitting

13	1. control different TVs in different rooms 2. music 3. photos in picture frames 4. parties, different music in different rooms; 5. superbowl party 6. would be very convenient for hotels to use to distribute movies & music to rooms
14	side yard wireless camera (currently hooked up to one TV) speakers in the kitchen & living room when I have company, play music timer: turning off TV, to regulate how much is watched day care: remote web cams to give sense of security remotely turn on TV/music for kids that don't nap
15	multiple activities: big house and everyone wants to do different things music in the kitchen see who's at the door kids watching TV access security cameras from office, maybe offer remote access to house music and different things in different rooms news
16	music library in any room films/dvds, move to any room camera outside (security) check on the kids speakers outside for a party
17	connecting laptop: videos & movies sound control (5.1, 7.1) lighting control office: presentations (show different things on different screens) "favorites"/playlists on living room speaksrs
18	stream music off computer pic from computer to living room videos from computer into bedroom big house- serious security monitoring have cable connected to computer, distribute throughout house open gate/door from within house family photos as screensaver on TV

What type of person is this for?

1	tech people or gadget people, once its been tested and the bugs are out, I buy it. Hooking up wires to TV , new radio can take time to learn. When upgrgaded to DSL, she did it. Maybe her brother who likes gadgets, friend's children, or friends' husband. Sems like its very expensive. technology is time consuming, it takes time to figure something like this out, need to read instructions, etc.
2	Normal people... she knows several high-tech types who'd like it. Key is NO monthly subscription, like slingbox, no subscription. During bowl season used PIP. All-in-ones good b/c no clutter
3	tech-savvy people. Sharper Image type stuff. People who are familiar with computers (enter, add, remove, click, copy, paste) "seems expensive" Mom, Dad, older peopple who have houses.
4	people who like gadgets, like to fiddle with things. Like me and all my friends. I have a friend with a bunch of TVs
5	Early adopters, someone wealthy, need to have a lot of rooms. Need to be a homeowner. Expensive to set up, doesn't work with roommates. No point in small apartment. He's in the second batch of adopters.
6	Not for everyone. A normal person might have a problem. Withn an hour or 2 I could do everything. Not for my father. The recipe stuff is a great idea, flexible. Lots of my friends, people who have larger houses.
7	Geeks and engineers, people into technology, people who work with semi-conductors. it's not user friendly--complicated & cumbersome people who work at home; people who don't go out too much (i.e., not me)
8	someone who likes to have cutting edge technologies (like me) people with kids who want to keep them entertained someone who shops at Frys or Best Buy, not walmart
9	(didn't ask)
10	average person, vast majority, more than 50% of the Bay Area. (includes me) not for cash-strapped people

11	those who are up on the technology and want to take full advantage (people like me) more affluent technology buffs
12	anybody (includes me) someone who lives in a large house someone who is handicapped, has limited mobility
13	if you're into cool gadgets like me, if you have redundant gadgets (>1 TV, etc.) lazy in a good way--people who watch TV, listen to music, use computers good for families--allow restrictions, monitors, controls
14	I'm not tech savvy but I would totally use this average person (middle class or high class) not my parents, they're not into computers, they would be intimidated 25+ years old
15	just about anybody older people would be intimidated kids & teens would love it parents personally-yes! (emphatic), also nearly all friends & family
16	people that have a lot of the latest electronics personally? Possibly (means no) people with bigger house, electronics in every room
17	anyone nowadays, we all have lots of devices and it should be easy to connect them
18	"ideally" 3 on a tech savvy scale of 1-5, (I'm a 4-5) personally? Possibly--60-70% of the time maybe not housewives

Recipe/Setup ranking discussion (after ranking first set of Recipes/Setups

1	Brother hooked up video camera for kitchen to watch mom, but if we had an intecom it wold be better for "helping out" from far away (giving instructions to mom and seeing what she's doing). Music to clean by, volume not too loud. Mom can't reason about machines, doesn't realize you have to press play every time. I could appear on the screen in her room. In her picture-in-picture and sat "press play."

2	husband would like double couch potato. Kids and pets, show 'em pictures. Get everything backed up on CD;s, tha way digital photos aren't all over the house. Lazy man's way to answer the door. Husband uses wireless laptop on the road (trucker) from coffeechops, communicates with cell phone and email. (he's a trucker). seems to do a fair amount with digital media (photos, especially) knows about DVR, SlingBox (but doesn't own them)
3	room monitor: I'd use it to snoop on my boyfriend very into music--currently has iPod, iPod speakers, iPod remote. Turns music on in several rooms (radio, presumably) to make it louder interested in several of the recipes tried video calls on PC before but too blurry and slow. Boyfriend would put weather #1. He likes news. Move music between car and house.
4	Better than PIP. Bought an LCD TV and old one not doing anything. Front door camera for mom.Shows off movies. Better than cell (for message at beep). Use it for Battlefield 2 (a squad game with VOIP communications) Upload videos of game play (captured using FRAPS) to a service like putfile.com Story about friend who accidentally posted embarrassing video of self on putflie.com
5	Has small place, more relevant for people with big houses. It feeds the "lazy factor"; "it would be cool" (by implication, perhaps, not essential)
6	Wife would want to see who's at the door. He hooked up X10 camera to a aham radio for this. Dobbler radar feeds? Cool! It would be one click weather. Rathe rhave music integrated into infra-red light sensors to detect presence. This would be faster for who's at the door.
7	Don't watch much YV, lots os DVD's. Its not necessarily better than how I do it now. 3 level house so intercome hndy.
8	Now checks traffic before going home. Radio and web for traffic. Wife likes music, photos.
9	Room monitor useful cuz has a 2 year old child Intercom--I hate screaming

Recipe ranking discussion for Recipes/Setups involving shared devices (after ranking second set of Recipes/Setups)

1 Doubts she'd use any, too "Big Brother." I don't think I'd use anything on the list now. Need some kind of firewall. Wouldn't want links into my house. So much invasion of privacy, if someone wants to hack in from office and see what you're doing. Isntant walkie-talkie maybe for mom, otherwise no.

2 Ausio letters cool, video phone call something she'd thought about but too expensive. Computer baseball.

3 Gas costs too much. Video phone for far away friends, keep you connected. Spy on boyfriend.

4 Neat feature

5 Sharing "my radio station" would be cool. Video phone call at party, a la drunk dialing.

6 parents older, always alone. Mom uses cordless phoe as medic alert. Put a camera in the garage so I can see if their car is gone. Father carries cell phone but won't turn it on unless he wants to make a call. Sis or brother tak to parents everyday. Push to talk would by-pass phone. Parents would rank the same. "videophones are long overdue"

7 There should be more face to face. Email is bad. Watching TV should be done in person, eat junk food together. Otherwise impersonal.

8 Wouldn't want to share. "I don't like them"

9 Reminds me of big brother, It's great, you get more involved video call makes the call more personal

Would the ability to share devices make this more useful?

1 No! Friends would have to have it to, and they wouldn't (too expensive)
Seems that this would invite invasions of privacy (the suggestion of sharing seemed to raise this specter for her)

2 Broader consumer audience, more people to audio letter.

3 Some excitement about spy camera, but no so much into making. it comes down to cost (?) Sharing means friends and family would have to invest, too. Not likely to happen.

4 Neat feature

5 Not crucial, maybe for special occasions (friend moved to Malaysia.

6 Yes, its valuable. "Now these are more interesting," (the second group of recipes.)

7 No. Doesn't make it more useful.

8 Not useful wouldn't do it. "unless you lived over seas or something." Wouldn't want to share.

9 Yeah, a lot more useful

Would recipe sharing be useful and what role would you play?

1 (didn't ask)

2 Share photos, music, play both roles/

3 Not so much of this.

4 concerned about viruses, malicious attacks downloading stuff from strangers. Firmware upgrade PSP attack

5 Recipes are easy to create, no need to share. More excited about sharing content than recipes.

6 Wouldn't use others, but would make them for people. Sharing would aleviate headaches with parents, easier for them to say "I want this." likely to be a producer, but wary that it's too hard

7 Not interesting.

8 Set up music or TV for wife.

9 Didn't ask.

What do you think of the form factor?

1 childproof, dropable, seams resaonable size. "You need to be able to read it."

2 smaller is better, cut it in half. Touch is OK

3 Like size of it. Too heavy to carry around (it would have to stay in one room). Won't lose it. Don't have to type on small keys. Easy to read.

4 Slightly too big for me, but my mom would need this size. could imagine using the Nintendo DS or similar device

. Had big problems with how hot the tablet got "burning my legs." Maybe too big, want to fall asleep watching TV with it on chest. Should be bigger than a Treo, bigger than a remote. Heavy too.

6 Heavy to hold. Has to have touch (like using a finger). Nice to have big screen.

7 Should be smaller, compact. Better yet, integrate it into certain appliances, attach it to a plasma TV, or wall.

8 Etch-a-sketch, too heavy and big. Should be a big remote. Better to have it in phone.

9 Palm pilot + etch-a-sketch

10 too large, heavy, bulky; phone is too small; maybe 1/2 the current size (easy to carry and not in the way)

too hot & heavy, fells like it would break if I dropped it
could be 1/2 the size (lose some of the whitespace)
Like to hang it on the wall, have it double as something else
(like another display)--then it's the right size
11 have one per room

hot & heavy, size is big but I like it, about the size of a
magazine (should be lighter though)
touchscreen was not responsive and didn't give adequate
12 feedback

just how it is--no bigger
13 would say it should be smaller but you already have to scroll

initially thought too big, but liked it in the end; lists & icons
were the right size
14 smaller than a laptop

dimensions are good, should be thinner
15 hang them on wall in key locations

little big, little warm
better: size of a playstation portable
1/2 or 1/4 the size
16 maybe even size of PDA or phone

17 too big, should be more like a remote control

18 too thick, should be only the size of the LCD

Would the tablet be yours or the households, and how many would you have?

Concern about conflicts--what if something is already
connected to the thing I want to use?
1 I would only have this because of my mom

2 Start out with one and see/

3 1 should do it

4 Just 1, in the living room

5 Household's. 2, I for living room, 1 for bedroom.

I'd have 1, wife would have 1. Want to be able to customize
it, like a dashboard. I'd like to have a lot of control. Go back
and get favorite things.Wife limits him to 1 TV in house, but
6 he has his own remote. 2 remotes, so not missing.

Unclear. See if there's a benefit, if it saves time. (start with
7 one per household, if any.)

8 Share with wife. Programs multiple things on the All For 1.

At least 2, one's the master so adult can supervise child. Need
9 child lock.

10 Maybe 1 per person, or just one with logins

11 1 per room; belongs to household

1 in bedroom, 1 in living room; belongs to household
12 maybe one in the wall in kitchen
13 1 for the house
(at first) 1 is sufficient
14 (after) well, maybe another one for the bedroom
want to have several, hang by the door
15 ones in different rooms
16 no need for more than one
17 1 in every room, owned by everyone

18 3 or 4 in the household with logins to control preferences

What do you think about the terms "recipe," ingredients, etc.?

1 Didn't record.
2 Didn't record.
Don't like it (offered this without being asked). Sounds like a
food thing. Should be "tasks" or something. "I mean, I get
3 it..."
It's OK, I mean, you have to name them something. Doesn't
add anything, though. Don't need the ingredients list, just see
4 it from the recipes
Seems hokey, maybe for a homemaker in the 50's or
5 something
Recipe didn't hit me right off the bat. Once I think about it, I
don't have a problem with it. For normal everyday people, its
6 comofrtable for non-tech people.
Recipe sounds like menu. Call it something more technical.
Call it devices. Maybe configuration or template would be right
7 for teachers (I.e., laypeople).

Weird name. Doesn't correlate with technology. Hate it.
8 (Spontaneously offered that he didn't like it, before we asked)
Too feminine. "Excuse me Bob, what's the recipe." (said
sarcastically) Egg and spatula. Really doesn't like it.
Particularly didn't like "prepare" ("Prepare doesn't mean
9 play"). Awkward and artificial.