Activity-Based Prototyping of Ubicomp Applications for Long-Lived, Everyday Human Activities

Yang Li[1] & **James A. Landay**[1, 2]

[1] Computer Science and Engineering
DUB Group, University of Washington
Seattle, WA 98195-2350
{yangli, landay}@cs.washington.edu

[2] Intel Research Seattle
1100 NE 45th Street, Suite 600
Seattle, WA 98105
james.a.landay@intel.com

ABSTRACT

We designed an activity-based prototyping process realized in the ActivityDesigner system that combines the theoretical framework of Activity-Centered Design with traditional iterative design. This process allows designers to leverage human activities as first class objects for design and is supported in ActivityDesigner by three novel features. First, this tool allows designers to model activities based on concrete scenarios collected from everyday lives. The models form a context for design and computational constructs for creating functional prototypes. Second, it allows designers to prototype interaction behaviors based on activity streams spanning time. Third, it allows designers to easily test these prototypes with real users continuously, *in situ*. We have garnered positive feedback from a series of laboratory user studies and several case studies in which ActivityDesigner was used in realistic design situations. ActivityDesigner was able to effectively streamline a ubicomp design process, and it allowed creating realistic ubicomp application prototypes at a low cost and testing them in everyday lives over an extended period.

ACM Classification Keywords

H.5.2 [User Interfaces]: Prototyping; D.2.2 [Design Tools and Techniques]: User interfaces.

Author Keywords

Ubiquitous computing, activity-centered design, rapid prototyping, *in situ* testing, context-aware.

INTRODUCTION

Ubiquitous computing (ubicomp) promises to support our everyday activities by weaving computing power into the fabric of everyday life [32]. However, designing for the complex, dynamic real world poses challenges that the well-understood, desktop environment does not [25]. There has been increasing interest in new design methodologies for ubicomp applications that involve rich, ever-changing contexts of everyday life [3, 4, 15, 18, 30]. In particular, Activity-Centered Design (ACD) has shown promise by encouraging designers to focus on evolving high-level human activities and by offering an activity-centric view of human-computer interaction [4, 15, 18]. ACD uses a larger unit (long-term, high-level activities such as staying fit) for analysis and design than simple tasks (such as using a treadmill), which have been the focus of traditional approaches. This allows rich everyday context to be systematically incorporated into designs and promises better results by orienting designs towards human needs.

ACD, however, is largely a set of perspectives and concepts. It is still an art to apply it in practice. In addition, there has been little work in developing this conceptual approach into a tangible ubicomp design process and providing computational tools to support it. This seriously hinders the application of a valuable methodology that has the potential to make a significant impact on ubicomp. Our goal is to support activity-centered ubicomp design by creating and experimenting with an *activity-based ubicomp prototyping* process (see Figure 1) and providing the appropriate tool support for that process. Activity-based prototyping combines the theoretical framework of ACD with traditional iterative design.

Research Challenges

The shift of focus from tasks to activities raises three new research challenges for activity-based ubicomp prototyping.

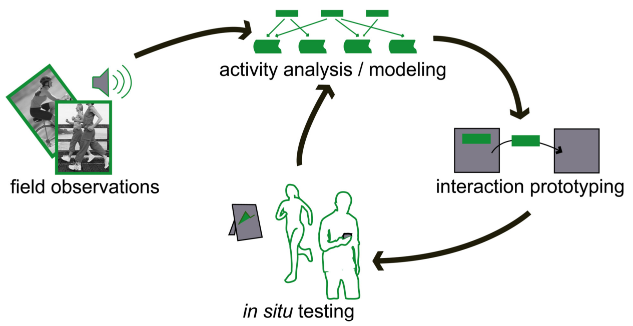


Figure 1. An Activity-Based Ubicomp Prototyping process. The skeleton of the process is based on the design/test cycle of iterative design. Note that activities are transformed from observations, to models, to computational constructs for functional prototypes.

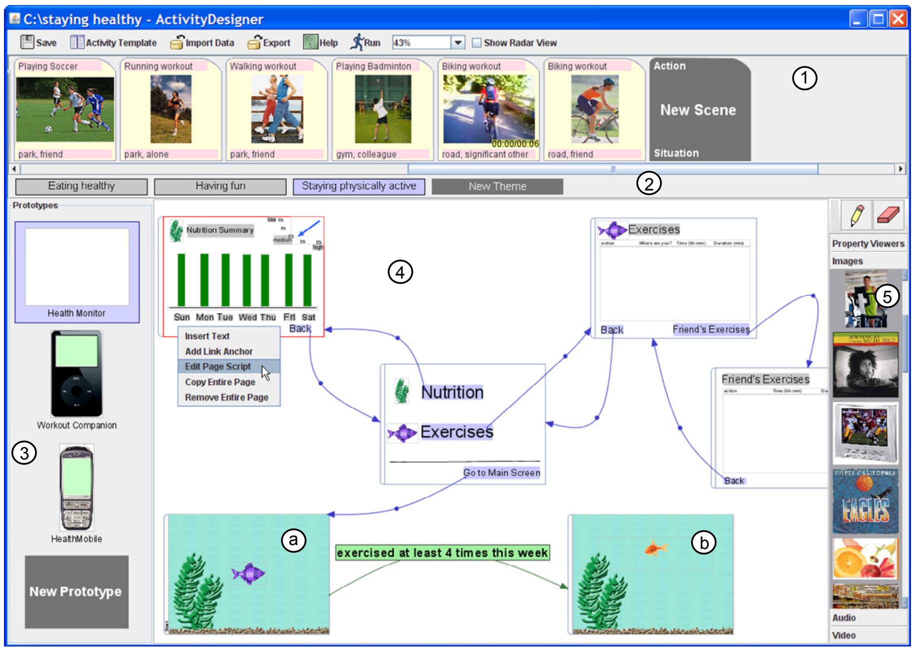


Figure 2. The ActivityDesigner design environment includes (1) Scene Panel that contains a collection of scenes representing everyday activities; (2) Theme Panel that contains multiple themes representing different aspects of everyday lives; (3) Prototype Panel showing the thumbnails of created prototypes; (4) Storyboard Editor for specifying the interaction logic of a prototype; and (5) Resource Panel that contains reusable dynamic interface components created via demonstration, images, audio clips and video clips.

From Everyday Observations to Activity Models

A design should be grounded in people's existing activities. It is important to include these activities in a prototyping process as a context and *resource* for design. A long-term, high-level human activity often evolves through multiple situations that include rich contextual information about people's everyday lives. The challenge lies in incorporating this diverse and complex information, which often comes from field observation, in a simple way. Previous tools have lacked a framework for representing contextual information and have not supported leveraging field observations.

From Activity Events to Activity Stream-Based Interactions

A high-level human activity (e.g., "aging in place" or keeping fit) spans an extended time period (e.g., months or years). Although proactive behaviors based on a single event may still play an important role (e.g., informing a care provider when an elder leaves her home), interaction behaviors for an evolving long-term activity must often account for the collective status of *a stream of individual events* (e.g., "how often has the elder exercised this week" or "has the user been eating healthy food this month?") In addition, inaccurate activity sensing can generate noisy activity event streams. Previous tools have not allowed designers to prototype stream-based interaction behaviors.

From Laboratory Prototypes to Real World Experiments

The impact of an application designed to support long-lived activities can only be observed when it is tested with real users in their everyday lives over extended periods of time [8]. However, previous tools mainly support testing prototypes in a controlled environment. This method is unrealistic and it makes it hard to observe how users' behaviors evolve in response to the new technology. Although recent work [5] has begun to enable *in situ* testing of prototypes, designs are limited to a constrained domain that only allows simple interactive behaviors. *In situ* evaluation is still challenging in general, as the realism and robustness required by such a test incurs a high cost for prototyping and deployment.

Our Contributions

We designed and realized an activity-based ubicomp prototyping process in the ActivityDesigner system (see Figures 2 & 8). It is the first tool that allows designers to leverage human activities as first-class objects for designing functional prototypes. This innovative tool also lets designers iterate on a design based on human activities, thus orienting a design towards users' high-level needs throughout the design process.

This major contribution is supported by three novel features in ActivityDesigner, which directly address the three challenges discussed previously.

- It allows designers to model activities by extracting them from concrete scenarios or field observations of everyday life, using a framework that we derived from Activity Theory [18] and our own fieldwork [13].
- It enables the rapid creation of prototypes based on modeled activities by allowing designers to specify stream-based interaction behaviors easily via direct manipulation and an activity query language.
- It supports real world experiments by generating prototypes that can run on various target devices for continuous *in situ* testing over an extended time period, and by allowing designers to monitor and analyze participants' behaviors for the next design iteration.

CONCEPTUAL EXPLORATION

To identify the key steps in and to design appropriate support for activity-based ubicomp prototyping, we explored both theory regarding Activity-Centered Design and existing design practice. Here we discuss how activity-based ubicomp prototyping and the design of ActivityDesigner are grounded in the results of this exploration.

The Theoretical Framework

Activity-Centered Design (ACD) [15] and Activity Theory (AT) [18], a descriptive theory grounding ACD, provide a consistent view for analyzing activity and offer a set of perspectives on design practice. A key idea from ACD is that a design process should address "…*how activities shape the requirements of particular tools* [here referring to ubicomp applications and devices] *and how the application of the tools begins to reshape the activity…*" [15]. This emphasizes the importance of leveraging existing human activities as a basis for design and addressing *evolving* activities in the design process. A prototyping tool should not only leverage user feedback on a tested technology in the next design iteration, it should also take into account the change in user behavior *due* to the new technology. These insights are reflected in the activity-based prototyping process that ActivityDesigner supports (see Figure 1).

AT uses an activity hierarchy to describe human activities. This hierarchy includes three key elements: activities, actions, and operations. An *activity* is the long-term transformation process of an object (e.g., a user's body) oriented toward a motive (e.g., keeping fit) [18], which is carried out by performing *actions* with each focused on attaining an immediate goal (e.g., taking medication to follow the doctor's instructions or running to burn calories). Actions can then be further decomposed into lower-level units called *operations* (e.g., opening a medication bottle).

This hierarchy captures the essence of long-term human activities, but is inadequate to describe the situational information associated with an activity. To represent context-rich human activities in a prototyping process, we extended this hierarchy by introducing three new concepts: situations, scenes, and themes (see Figure 3). In reality, actions (e.g., running) are often performed in certain circumstances (e.g., in a gym or park and with friends or alone), which we call *situations*. We call the combination of an action with its associated situation a *scene*, which represents a real scenario or observation of everyday life.

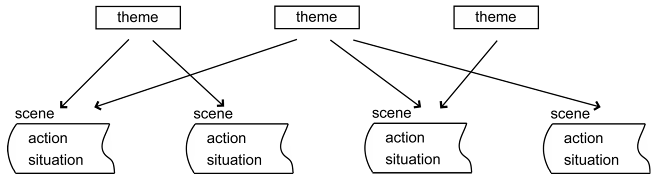


Figure 3. The conceptual framework for representing human activities in an activity-based prototyping process.

Because a particular action often supports attaining the motives of multiple activities (e.g., playing soccer may support both "keeping fit" and "having fun"), an AT style activity can be considered a specific view on a set of actions. Here we use the term *themes* to capture this particular meaning of "activity." Themes embody the practical use of AT's "activity" for ubicomp design, i.e., enabling an integrated view over individual observations and providing a broader unit for design. For clarity, in the rest of the paper, we use *activity* to refer to its generic meaning of what people do, rather than activity as defined by AT. The extended AT shaped our representation of human activities in the ActivityDesigner system.

Fieldwork

To supplement our knowledge of theory with learning from current ubicomp design practice, we conducted interviews with eleven professional designers [13]. Here we summarize the findings that influenced our support for activity-based prototyping of ubicomp applications. Most designers in our study relied on the rich user experiences and contextual information that is observed in everyday life. This information was often collected through field research (e.g., ethnography). It is important to bring these field findings into a design process so that designers can acquire a "context for design" [3]. The study also revealed the important role that media-rich representations (e.g., photos) played in the early stages of ubicomp design both to visualize this context and to brainstorm on design ideas.

In addition to the technical challenges faced by designers (e.g., dealing with brittle ubicomp technologies), our study also revealed the high economic and social barriers for experimenting with new technologies and communicating with other stakeholders (e.g., field researchers, developers and target users). As such, one key goal for our work is to bridge the gap among field research, design, and testing.

APPLICATION DOMAIN

Instead of targeting general ubicomp applications, we focus here on the design of a class of application behaviors that best manifest the challenges that we discussed earlier. This allows us to concentrate on the research issues of activity-based ubicomp prototyping, rather than being distracted by other issues such as the various types of output that ubicomp applications might have.

We focus on an important class of applications that improves our awareness of our own activities as well as those of others. For example, interactive digital picture frames have been built to give the family members of an elder a continued awareness of the elder's everyday status and help them to coordinate care activities [9, 28]. Likewise, applications have been built to encourage physical activity by visualizing and sharing people's workout status [7, 10, 22]. These applications leverage everyday activities as well as the context in which they are carried out over an extended period of time. They have shown great potential to improve our lives but currently require a huge amount of effort and time to design, develop, and deploy before they can be tested with end users in their real lives. This makes it difficult and expensive to explore this design space. We use this class of applications to validate our solutions to the research challenges we listed.

PROTOTYPING WITH ACTIVITYDESIGNER

In this section, we describe how the activity-based ubicomp prototyping process is carried out in ActivityDesigner. We focus on the three key features of ActivityDesigner that directly address our three main challenges. We use the example of designing for "staying healthy," which involves modeling everyday activities for staying healthy, creating application prototypes for supporting these activities, and testing these prototypes *in situ*. These design examples are not intended to show the best way of addressing this application domain. Instead, we demonstrate that designers can quickly iterate on different design alternatives by prototyping and testing with ActivityDesigner. We also show how activity-based prototyping is grounded in human activities throughout the design process.

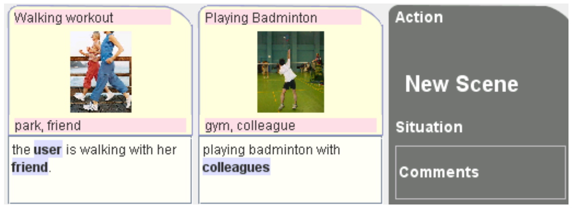
Creating Activity Models from Everyday Observations

A designer starts the activity-based prototyping process by modeling target activities based on everyday observations. This establishes both a design context and the computational structures for creating functional prototypes.

Representing Everyday Observations as Scenes

ActivityDesigner allows a media-rich representation of everyday observations. A designer adds concrete scenarios about everyday life to her design as *scenes*, which are arranged horizontally at the top of the screen (see Figure 2-1). A scene has four parts:

- the *action* performed in the scene, represented by the textual label at the top;
- the *media* (images or video clips) in the middle, which represent the scene visually;
- the *situation* in which the action is performed, briefly described by the textual label at the bottom;
- comments (unstructured textual notes), displayed at the bottom only when they are expanded (see Figure 4).



Walking workout

park, friend

the user is walking with her friend.

Playing Badminton

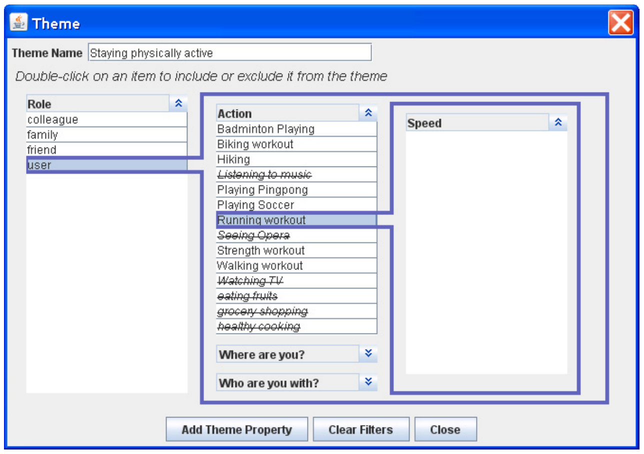
gym, colleague

playing badminton with colleagues

Figure 4. Expanded scenes. The comments for each scene are shown at the bottom with roles highlighted in blue.

Designers can add a single scene by clicking on the New Scene shadow. Designers can also automatically import a set of data into a new collection of scenes. To simplify the addition of field data, which might be collected in various forms, ActivityDesigner makes few assumptions about the structure of the data. By making scenes explicit and visible in the design, it helps designers to imagine usage scenarios and orient their design decisions towards real user needs.

Next, designers describe the stakeholders involved in a target domain and their activities of interest. In ActivityDesigner, we use the term *role* to refer to a stakeholder. A role (such as “elder” or “nurse” in the elder care domain) is a character used for constructing prototypes and is the part that a participant is going to play in an *in situ* study. ActivityDesigner can automatically extract roles by first identifying all the nouns in the scene’s textual data using a part-of-speech tagging tool [29] and then analyzing whether each noun represents a “person” using WordNet [33]. The roles are highlighted in a scene’s comments (see Figure 4). Designers can tailor the list of possible actions



Theme

Theme Name: Staying physically active

Double-click on an item to include or exclude it from the theme

Role
colleague
family
friend
user

Action
Badminton Playing
Biking workout
Hiking
~~Listening to music~~
Playing Pingpong
Playing Soccer
Running workout
~~Seeing Opera~~
Strength workout
Walking workout
~~Watching TV~~
~~eating fruits~~
~~grocery shopping~~
~~healthy cooking~~

Where are you?

Who are you with?

Speed

Figure 5. Designers create a theme by excluding irrelevant actions or situations in a tree view of the activity structures constructed from the created scenes (e.g., “eating fruit” is irrelevant to the “Staying physically active” theme).

and situations for each role (e.g., *biking* may not be a possible action for an elder) or add detailed properties to a model (e.g., adding a *speed* property for *running*).

Representing Aspects of Everyday Life as Themes

Designers can then create higher level structures, or themes, based on the activity structures reflected by the scenes (see Figure 5). Designers can create multiple themes to represent different aspects of everyday life for a domain (e.g., eating right and having fun are two aspects to staying healthy).

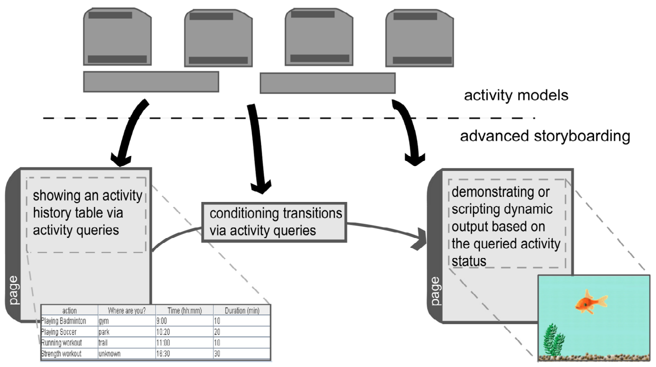
Created themes are arranged horizontally in the Theme Panel (see Figure 2-2). After selecting a theme, ActivityDesigner displays only the subset of scenes that is relevant to that theme. A scene can appear in multiple themes, and is not displayed if its action or any of its situation properties is excluded by the current theme.

As a design construct, themes add a layer between the overall goal of a target design domain (e.g., staying healthy) and concrete everyday scenarios that help achieve this goal. The use of themes offers several unique benefits. First, for structuring design context, themes allow designers to manage a large number of scenes easily by showing only a relevant subset of scenes. Second, for making activity models more operational for prototyping, themes can be used as filters at design or run time for identifying relevant events, such as showing all actions about having fun.

Creating Activity Stream-Based Interaction Behaviors

Designers can then create functional prototypes based on the activity models using a combination of techniques, including storyboarding, demonstration, and scripting (see Figure 6). ActivityDesigner allows multiple prototypes to be created at the same time (see Figure 2-3). This permits designers to think globally in terms of how each should behave to provide coordinated support for target activities.

Here we focus on the unique challenge of prototyping for activity-stream-based interaction behaviors. Activity models describe a single event (e.g., running) but not how a


activity models
advanced storyboarding
showing an activity history table via activity queries
conditioning transitions via activity queries
demonstrating or scripting dynamic output based on the queried activity status
page
page

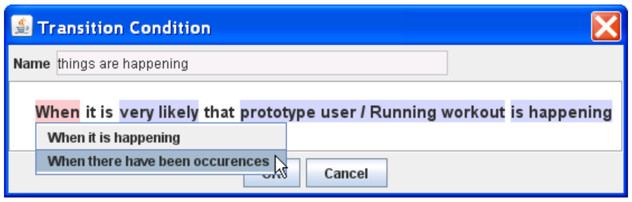
action	Where are you?	Time (hh:mm)	Duration (min)
Playing Badminton	gym	9:00	10
Playing Soccer	park	10:20	20
Running workout	trail	11:00	10
Strength workout	unknown	16:30	30

Figure 6. A storyboard page represents the interface output and a link between pages represents a transition that can be triggered by the user's input (e.g., a mouse click) or activity events. The status of activities, accessed via activity queries, can be shown as a history table or dynamic graphical output on a page, or used to condition transitions.

high-level activity status property should be constructed over time (e.g., how often did a user exercise this week?)

Activity event streams can be *multi-dimensional* (e.g., what people do and where they do it) and *uncertain* due to the inaccuracy of sensing devices and inference algorithms. It can be difficult for designers to specify queries such as "show all the user's workouts this week lasting more than 20 minutes" or "how likely is it that the user has been running today," since database query languages, such as SQL, require high levels of technical expertise to use. We designed an activity query language based on an analysis of queries needed by existing ubicomp applications as well as on feedback from professional designers.

As an example, consider a designer creating a Health Monitor application to improve the user's awareness of her nutrition (visualized via the growth of a plant) and workout status (visualized via the activity of a fish) (see Figure 2). To turn the purple fish (see Figure 2-a) into a swimming goldfish (see Figure 2-b) when the user has run enough for the week, a designer drags the "Running workout" action from the Scene Panel (see Figure 2-1) to the link from page a to page b. This creates a default condition for the link transition to take place (see Figure 7a).



Name things are happening

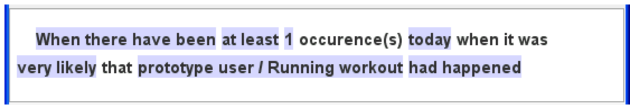
When it is very likely that prototype user / Running workout is happening

When it is happening
When there have been occurences

Figure 7a. The interface for specifying a transition condition. Each of the highlighted terms can be selected to change that piece of the condition.

An activity query is structured as an English sentence with customizable parts highlighted in blue. These parts may include activities of interest (e.g., prototype user / Running workout), a time period of interest, a number of occurrences, a certainty threshold (e.g., likely vs. unlikely) and a minimum duration for the queried assertion to be true.

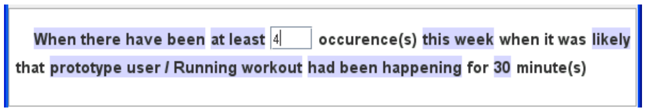
Designers can easily customize a query by changing the parts in blue. For our example, a designer is interested in whether there have been enough running workouts this week rather than whether the user is running when the condition is evaluated at runtime. The designer clicks on "When" which brings up the list of options in Figure 7a, and selects "When there have been occurrences." This causes the query to be restructured to Figure 7b.



When there have been at least 1 occurence(s) today when it was very likely that prototype user / Running workout had happened

Figure 7b. The restructured query of Figure 7a.

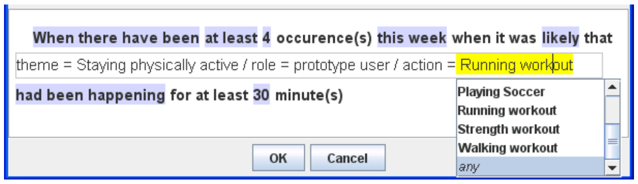
The designer can keep customizing the query to a desired form. Here, the designer changes the time period of interest to "this week," lowers the uncertainty threshold to "likely" and adds a minimum duration of 30 minutes by changing "had happened" to "had been happening for a certain duration" (see Figure 7c).



When there have been at least 4 occurence(s) this week when it was likely that prototype user / Running workout had been happening for 30 minute(s)

Figure 7c. The query for the condition testing whether the prototype user has run enough in a week.

A designer can also modify the queried activity by clicking on it (e.g., "prototype user / Running workout" in Figure 7c). This activates a text field with the queried activity represented in its detailed form (see Figure 7d), which is a list of filters connected by slashes. To make the results conditional on the number of any kind of workout rather than only on running, a designer edits the default filters by adding "theme = Staying physically active" and changing "action = Running workout" to "action = any." This will allow any actions of the "prototype user" included in the "Staying physically active" theme that satisfy other parts of the query to be counted at runtime.



When there have been at least 4 occurence(s) this week when it was likely that
theme = Staying physically active / role = prototype user / action = Running workout
had been happening for at least 30 minute(s)

Playing Soccer
Running workout
Strength workout
Walking workout
any

Figure 7d. Specifying a set of activity filters.

As a designer types in the text field, auto completion shows the most likely candidates based on the activity models, and prunes the candidate list based on the properties that have been added so far. For example, when the caret is on "running workout," the candidate list shows all actions filtered by both "Staying physical active" and "prototype user." The "prototype user" is interpreted at runtime using the role assigned to a prototype. Designers can also use a Boolean expression for a filter, e.g., a designer can specify queries for all the running workouts of the *user* or her *friend* in the gym or park using the following filters.

user | friend / running workout / gym | park

By providing explicit support for uncertainty, designers have the flexibility to address design situations that may require different levels of confidence in sensed data. For example, for emergency events like an elder falling, a UI behavior should probably be triggered even when it is only 50% likely. In addition, designers can also specify a condition to be only true when things are unlikely to be happening (e.g., when it is unlikely that a user is driving).

Testing Prototypes Locally or *in Situ*

Testing is as important to activity-based prototyping as to traditional iterative design. Here we first describe how ActivityDesigner allows local testing at any time for debugging and then describe the tool support for testing these prototypes *in situ* for real world experiments.

The Testing Workspace

Designers can test one or more prototypes at a time in the Testing workspace (see Figure 8). A designer starts testing a design by loading an existing test session (for monitoring an ongoing experiment or replaying recorded activity events) or by creating a new session on either the local machine or a remote server for an *in situ* experiment. For a new session, ActivityDesigner generates the associated database structures for storing activity data.

Once a session is started, the green time cursor in the Log Viewer (see Figure 8-1) moves based on a designer specified speed, which can be in real time or at a faster rate. The Prototypes Panel (see Figure 8-2) shows the prototypes being tested. They are replicas of what end users will see and interact with. A designer can interact with these prototypes as an end user can. A designer can simulate sensor input in the Activity Simulator (see Figure 8-3). Simulated events are streamed into the associated database of the current session, against which the queries issued by application prototypes will be processed.

The Activity Simulator's interface is dynamically structured based on the activity models of the design being tested. For example, the action list is reduced to show only the possible actions of the current role. This helps reduce the wizard's effort when there are many items in a list. To simulate noisy sensor input during a Wizard of Oz (WOz) test, a designer can specify how accurate the sensor input should be. The ActivityDesigner will then generate random errors based on the specified accuracy (e.g., inserting a walking event when a designer specifies that the user is running). This can help designers uncover design issues related to the accuracy of sensing and encourage them to think about exceptions.

Designers can also analyze participants' behavior patterns employing the same activity queries used for design. For example, based on the query and time granularity specified, ActivityDesigner can generate a graph that shows how the number of instances of queried activities changes over time during a test. It also allows designers to export the data as a text file for further analysis in statistical tools.

Deploying a Design to the Field

There are two major tasks required to deploy an ActivityDesigner design to the field: installing prototypes on target devices and incorporating appropriate activity sensing components. ActivityDesigner provides integrated support for these two tasks.

In deploying a prototype to a target device, ActivityDesigner offers two options. For high-end devices such as Tablet PCs, ActivityDesigner provides a virtual machine for running an ActivityDesigner prototype. For low-end devices that are not able to run the virtual machine, ActivityDesigner automatically transforms a prototype into an HTML+JavaScript-based web application using the Google Web Toolkit. The prototype can then run wherever a web browser is installed (e.g., Opera Mobile on a phone).

ActivityDesigner can easily incorporate various activity sensing components as these components communicate with the runtime via HTTP. In particular, ActivityDesigner supports two types of sensing components: those based on "human sensors" by which user participants self-report their activities throughout the day and those that automatically detect users' activities based on real sensors and inference.

ActivityDesigner provides both a mobile phone-based *in situ* journal and a Web-based interface for user reporting. ActivityDesigner also offers several real sensor-based components, including ones for the Mobile Sensing Platform [6], RFID-based activity sensing [26], and GSM/WiFi-based localization [20]. Based on the modeled activity structures, ActivityDesigner generates an XML profile describing each role's possible activities. The profile

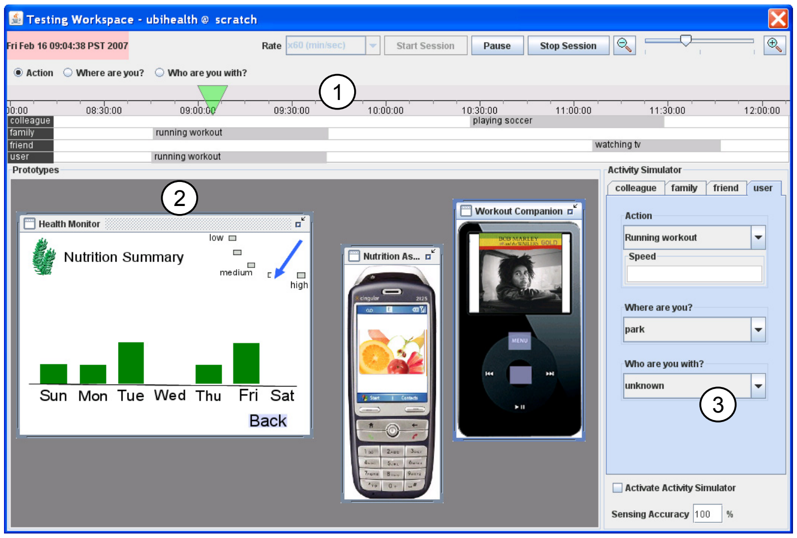


Figure 8. The Testing Workspace has three parts: (1) the Log Viewer visualizes the aggregated activities of all the roles along various dimensions; (2) the Prototype panel contains the prototypes being tested; (3) the Activity Simulator is used for simulating activities.

is used to generate the *in situ* activity journal and the web-based activity reporting UI. It also converts sensing results to a format understood by the ActivityDesigner runtime.

These two types of activity sensing components can be used in combination. Designers can specify which properties will be reported by a human participant and which will be detected by sensing modules for a generated profile. For example, a designer may specify that the "where are you?" property is to be detected by a location tracking system and the "who are you with?" property is to be self-reported.

IMPLEMENTATION

The ActivityDesigner system is implemented primarily in Java SE 6. The tool currently has 545 Java classes in about 40,000 lines of source code. We embedded BeanShell [2] into ActivityDesigner's runtime environment to interpret any JavaScript that designers might use for expressing more complex behaviors (e.g., making the fish swim in the example application). The *in situ* journal tool for reporting activities on a mobile phone is written in 500 lines of Windows Mobile C# source code. The web services for *in situ* testing are implemented as Apache Tomcat servlets.

Querying over temporal noisy data streams is still an evolving area [1] and there is no off-the-shelf support available. ActivityDesigner employs a simple but effective solution for activity stream processing. Our query engine is implemented as a wrapper over Xindice [34], an XML database we use as the data store for activity data streams. To deal with noisy data and measure uncertainty, we use a voting mechanism over a query-defined time window that calculates the percentage of time that the queried condition holds. Although this does not give a probabilistic measure of uncertainty, it gives a simple but cost-effective measure that is sufficient for testing prototypes.

USER STUDIES

We ran a series of laboratory user studies to find out how designers responded to the concepts of the ActivityDesigner and to gain their feedback for improving the tool.

Acquiring Initial Feedback with a Laboratory User Study

We first tested an early version of the ActivityDesigner with 10 participants (seven male, three female, Age: 20-43). This qualitative study explored which parts of the tool were understandable and where more research was needed.

Our participants had varied backgrounds: Technical Communications (three, one was also a professional UI designer), Information Science (one) and Computer Science (six, three of whom were undergraduates). All of the participants had general UI design experience. Two of them also had ubicomp development experience. Participants received a $40 gift certificate for participating.

Each experimental session took about two hours. The participants were asked to create scenes based on a set of data (video and images) prepared by the experimenter, and design three prototypes (two for elder care and one for keeping fit), ranging from simple to complex. Afterwards, we asked our participants to comment on the concepts and design of ActivityDesigner by answering a questionnaire.

Successes

All of our participants were able to finish most of these tasks with little assistance. They were helped by the experimenter when distracted by system bugs (e.g., refresh problems). Overall, our participants were able to understand the concepts and easily grasp most features of ActivityDesigner. ActivityDesigner was rated 6 (Std = 1.2, 1: not useful, 7: useful) for its overall usefulness, and 5.5 (Std = 1.1, 1: hard to use, 7: easy to use) for its interaction and UI design. As one participant commented:

> *…it was VERY intuitive and very easy to use, and using it to design is quick enough that one could use it for brainstorming as well as developing testable prototypes…*

Our participants easily understood the major concepts of the tool such as scenes, actions and situations as well as their use in design (Mean=6.2, Std=0.8, 1: hard to understand, 7: easy to understand). None of our participants had an Activity Theory background. One participant commented:

> *…it gives you a broad overview of all the actions and scenarios… it is useful so that you can picture the activities in your mind…*

Problems Revealed

Two major issues were uncovered in this initial study. First, participants who had a strong technical background felt they needed more control. For example, one participant gave a low rating, 3 (1: not expressive, 7: expressive) on the expressiveness of the early version of ActivityDesigner, because he focused on issues such as sensor attributes. As he commented:

> *…you can do a lot with images, audio, and video, but there's probably some design considerations that aren't represented as well as you might want – for example sensor reliability, failure model, etc.*

Second, we observed that the major difficulty participants encountered was in using storyboards to aggregate activity status properties—a feature provided in the early version of the system. Storyboards are effective for describing interaction logic but seemed less intuitive in manipulating query variables (e.g., counting occurrences of an activity). These observations motivated us to design a new activity query language in the current version of ActivityDesigner.

A Focused Study on the Activity Query Language

We then ran a user study to evaluate the activity query language with six participants (three female, three male, Age: 28-44). Our participants were representative of the target users of ActivityDesigner: one ubicomp researcher, one HCI researcher, two professional interaction designers, and two students who had interaction design experience. Only one participant, the ubicomp researcher, had a computer science background.

Each experimental session took about an hour. Participants were given a $30 gift certificate for their participation. We asked participants to specify activity queries in response to 15 questions prepared by the experimenter in natural English, e.g., "Has the user been in the classroom or theater for five minutes?" or "Did the user go to the gym this week?" Participants finished most tasks without any assistance and half of the participants needed some assistance for the first five questions.

Wisely, our participants reduced the need for customization by searching for an activity model that was close to the given question. For example, for the question "Is the user running with her friend at the park?" some participants picked the scene (see Figure 2)

"[user doing] walking workout [with her] friend [at the] park"

and then simply replaced *walking workout* with *running workout* in the query specification interface. We also observed that themes were heavily used by our participants to quickly locate an appropriate model in the Scene Panel.

We received positive feedback on the activity query language for its understandability (Mean=6.2, Std=1.2, 1: difficult to understand, 7: easy to understand) and ease of use (Mean=6.7, Std=0.5, 1: difficult to use, 7: easy to use). In terms of the expressiveness of the language, the participant who had a ubicomp background commented that

…*This allows you to easily specify a wide range of queries about people's everyday activities*…

Two of six participants knew SQL and commented that the activity query language required much less effort in comparison with that of SQL.

The main complaint seemed to be that there were not enough options for specifying times. This observation will help us revise the current design of the activity query language.

CASE STUDIES

To further validate the usefulness of ActivityDesigner, we have used the tool in our own research and released the tool for use in design projects conducted by other researchers.

Case Study #1: Health Monitor

To gain firsthand experience of designing with ActivityDesigner, we focused on the fitness domain and worked through an entire process of field data collection, design, and *in situ* testing. We collected data with six participants over two weeks using an ESM tool [14] and a diary method to understand existing practices for keeping fit. We collected rich data that embodied two main themes: eating healthily and staying physically active. Next we imported the field data into the tool as a basis for design. Then we created prototypes based on the modeled activities. In particular, we designed a Fitness Fish Display that gives a user an ambient view with different types of fish to represent different physical activities that a user performs for keeping fit, as well as a detailed view of these activities. This application was also inspired by two existing applications created by others: Fish'n'Steps [22] and UbiFit [10]. We then deployed the design with two participants for one week, using two Tablet PCs as target devices. Our participants brought these devices into their homes.

To examine the feasibility of web-based *ex-situ* and mobile-phone-based *in situ* journaling for simulating activity sensing, we asked each of our two participants to try a different journaling method. Our participants reported their workout activities, durations, locations, and co-located people via the journaling interfaces generated by ActivityDesigner.

To learn how ActivityDesigner could support the design of collaborative activities that involve multiple users operating multiple applications, we designed a second version of the application, the Collaborative Fish Display. This was also in response to one participant's comment "…*it is kind of boring to only be able to see my own status on the screen*…" We deployed it with the two participants for an additional two weeks. The status of both participants' activities is visible on a Fish Display with each represented as a separate fish. The more workouts a participant has done, the higher his fish swims and the bigger it grows. One participant commented that "*...Although the prototype doesn't automatically detect my activities at the moment, it is already very useful to be able to visualize my journal...*"

It took one of the authors less than a day to design and deploy each of the two prototypes. In the course of the study, we were able to monitor these deployed devices as well as the users' physical activities via the Testing Workspace. The major problem we found with the deployment was the stability of our runtime environment. Occasionally, the prototype didn't respond and our participants had to restart it. We found that ActivityDesigner gave us the leverage to explore and structure a design space quickly by prototyping and testing ideas before devoting any effort in developing a fully engineered system and incorporating sensing technologies.

Based on this study, we designed a new application, Health Monitor, that shows both the nutrition and workout status of a user and her friend (see Figure 2). We also designed a mobile version of the application and tested it on a mobile phone browser. In addition, we used the Mobile Sensing Platform [6] to detect the user's physical activities and self-report methods for users to report their eating activities.

Case Study #2: UbiGreen

ActivityDesigner has been used in designing UbiGreen, a transportation activity display to encourage environmental stewardship. This was a project conducted at an industry research lab. (Note that one of the authors was involved in this project and he was the only person on the team who had a computer science background.) By displaying transportation activity information on the background of an individual's mobile phone, UbiGreen gives people an ambient awareness of their weekly transportation usage. By making people more aware of their transportation patterns and by giving them positive feedback when these patterns have less impact on the environment (e.g., walking, biking, or taking public transit), UbiGreen helps people make personal changes that can improve our global environment.

The researchers in this project started their design process by collecting field observations about people's everyday transportation activities. They then imported the data into the ActivityDesigner and created a mobile phone prototype of UbiGreen. The researchers created a testable prototype in under five hours, more than half of which time was spent on documentation problems and program bugs. Our main conclusion from this study was that some field data does not use generic role names (such as drivers). Instead, it uses concrete names of people (such as Bob), which currently cannot be extracted automatically by ActivityDesigner.

Case Study #3: Social Garden

ActivityDesigner has also been used in Social Garden, an independent project with which none of the authors were involved. The goal of the project was to design a web-based ambient display for students, staff, and faculty in a university department to improve their social awareness. Social Garden visualizes the department's overall social activities as well as those of individuals using a garden metaphor, e.g., blooming flowers indicate social status.

The researcher in this project began the design process by first determining what social occasions might occur in the department, including seminars, student-advisor meetings, or hallway chats. In less than three hours he had created a prototype in ActivityDesigner. As a next step, the department's existing RFID sensing infrastructure will be linked with the ActivityDesigner runtime to automatically detect targeted social occasions.

DISCUSSION & FUTURE WORK

In this section, we discuss how well we have answered the research questions we list at the beginning of the paper with ActivityDesigner, as well as our plan for future work.

Limitations

Among the limitations of the ActivityDesigner system and framework is a lack of hierarchy for actions. For example, traveling to an unknown location can be decomposed into printing out a map and driving to the place. We intend to support action hierarchy in the future. Also, our framework does not provide explicit support for modeling exceptional events, e.g., the elder falls. Conceptually, these events are often not performed by users intentionally and are thus not performed to attain a motive in the AT sense, although ActivityDesigner does not prohibit including these events.

Lowering the Floor and Raising the Ceiling

Based on our laboratory studies as well as the three case studies, we believe ActivityDesigner significantly lowers the floor for designers to create testable application prototypes that address long-lived human activities. Applications such as UbiFit [10] and the CareNet Display [9] previously took *teams of designers and engineers weeks or months* to design and develop before they could be handed to end users for long-term *in situ* studies. With ActivityDesigner, it took *a designer* (one of the authors) *less than a day* to create and deploy a testable prototype of each of these applications. This success is mainly due to the effective framework that ActivityDesigner is built on and the integrated support for designing and testing that it provides.

Though we still need more long-term evidence of the benefits of ActivityDesigner, we have already seen positive results. In addition to recreating several existing applications supporting long-lived activities, ActivityDesigner has also been used to create and test several new applications such as those reported in the case study section. ActivityDesigner also provides advanced designers with tools such as scripting for prototyping more complex behaviors. In addition, ActivityDesigner speeds up iterative design and empowers designers who have fewer technical skills to design for this sophisticated domain.

We plan to conduct longer and more extensive studies to investigate how ActivityDesigner, as well as the proposed design process, is adopted by designers in practice. In particular, we intend to find out how ActivityDesigner can increase designers' creativity, foster new ways of thinking and raise the ceiling for creating novel applications.

RELATED WORK

Ubicomp is too broad a domain to address in a single piece of work. Commercial tools, such as Director [24], allow designers to rapidly prototype GUIs. However, these tools do not provide the concepts and the necessary tool support for prototyping ubicomp applications. Previously, there have been efforts to provide tool support (e.g., [16, 21, 23]) for designers to create early prototypes of ubicomp technologies, or for end users to configure interaction behaviors of a pre-deployed sensor environment (e.g., [11, 12, 31]). Similar to this prior work, ActivityDesigner is intended to allow designers who have a less technical background to prototype ubicomp applications rapidly.

ActivityDesigner, however, addresses three new challenges raised by activity-based ubicomp prototyping that are not addressed by prior tools. First, ActivityDesigner is the only prototyping tool that explicitly supports modeling human activities based on field data. This feature is important as it bridges the gap from collecting individual observations about everyday life to creating the structures necessary for creating functional prototypes. We designed this support based on both a thorough exploration of the theoretical framework of Activity-Centered Design [15, 18] and on investigations of existing ubicomp design practices [13].

Second, ActivityDesigner allows designers to design based on a stream of events that span an extended period of time, which is far different from designing interactive behaviors based on a single event that occurs in the moment, as previous tools have done. This distinction is important as activity-based prototyping addresses long-term, evolving activities. Our query language gives designers the flexibility to specify queries of activities of interest that address sensing uncertainty and time constraints in a simple way.

Third, in contrast to prior work that focused on laboratory studies, ActivityDesigner enables low-cost prototypes to be tested continuously *in situ*. Our testing methodology benefited from traditional Wizard of Oz approaches that reduce the cost for testing by avoiding the requirement of sensing infrastructures [16, 19, 21, 23, 27]. We also benefited from the Experience Sampling Method (ESM) and diary studies that extend WOz approaches for testing prototypes *in situ* [5, 17]. However, ActivityDesigner enables activity-based interaction behaviors that are more complex than the behaviors supported by the prior work. In addition to generating a set of fully interactive prototypes, ActivityDesigner automatically generates an activity database structure for managing activity streams and processing queries. ActivityDesigner inherently supports multi-user experiments where each participant plays a different role. It also allows designers to monitor a test over time and analyze participants' behavior patterns.

CONCLUSIONS

This paper presented an activity-based ubicomp prototyping process and its tool support. ActivityDesigner is the first tool that allows designers to leverage human activities as first class objects for design. ActivityDesigner garnered positive feedback in a series of laboratory user studies and case studies. It provides a useful platform for exploring Activity-Centered Ubicomp Design, which opens new ground for ubicomp design. ActivityDesigner has been released at http://dub.washington.edu/activitydesigner.

REFERENCES

1. Abadi, D.J., et al., The Design of the Borealis Stream Processing Engine. In Second Biennial Conference on Innovative Data Systems Research: CIDR 2005.
2. BeanShell, http://www.beanshell.org/.
3. Beyer, H. and Holtzblatt, K., *Contextual Design: A Customer-Centered Approach to Systems Designs*. 1997: Morgan Kaufmann.
4. Bodker, S., *Through the Interface: A Human Activity Approach To User Interface Design*. 1990: Lawrence Erlbaum Associates.
5. Carter, S., et al., Momento: Support for Situated Ubicomp Experimentation. In CHI'07, pp. 125-134.
6. Choudhury, T., et al., The Mobile Sensing Platform: An Embedded System for Capturing and Recognizing Human Activities. To Appear in *IEEE Pervasive Computing, Special issue on Activity-Based Computing*, Apr-Jun 2008.
7. Consolvo, S., et al., Design Requirements for Physical Activity-Enabling Technologies. In CHI'06, pp. 457-466.
8. Consolvo, S., et al., Conducting In Situ Evaluations for and with Ubiquitous Computing Technologies. *International Journal of Human-Computer Interaction*, 2006. 22(1): pp. 107-22.
9. Consolvo, S., et al., Technology for Care Networks of Elders. *IEEE Pervasive Computing: Special Issue on Successful Aging*, 2004. 3(2): pp. 22-29.
10. Consolvo, S., et al., Activity Sensing in the Wild: A field trial of UbiFit Garden. To Appear in CHI 2008.
11. Dey, A.K., et al., a CAPpella: Programming by Demonstration of Context-Aware Applications. In CHI'04, pp. 33-40.
12. Dey, A.K., et al., iCAP: Interactive Prototyping of Context-Aware Applications. In Pervasive'06, pp. 254-271.
13. Dow, S., et al., External Representations in Ubiquitous Computing Design and the Implications for Authoring Tools. In DIS'06, pp. 241-250.
14. Froehlich, J., et al., MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones. In MobiSys'07, pp. 57-70.
15. Gay, G. and Hembrooke, H., *Activity-Centered Design: An Ecological Approach to Designing Smart Tools and Usable Systems*. Acting with Technology. 2004: The MIT Press.
16. Hartmann, B., et al., Reflective physical prototyping through integrated design, test, and analysis. In UIST'06, pp. 299-308.
17. Iachello, G., et al., Prototyping and Sampling Experience to Evaluate Ubiquitous Computing Privacy in the Real World. In CHI'06, pp. 1009-1018.
18. Kaptelinin, V. and Nardi, B.A., *Acting with Technology: Activity Theory and Interaction Design*. 2006: MIT Press.
19. Klemmer, S.R., et al., SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. In UIST'00, pp. 1-10.
20. LaMarca, A., et al., Place Lab: Device Positioning Using Radio Beacons in the Wild. In Pervasive'05, pp. 116-133.
21. Li, Y., et al., Topiary: A Tool for Prototyping Location-Enhanced Applications. In UIST'04, pp. 217-226.
22. Lin, J., et al., Fish'n'Steps: Encouraging Physical Activity with an Interactive Computer Game. In Ubicomp'06, pp. 261-278.
23. MacIntyre, B., et al. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In UIST'04, pp. 197-206.
24. Director, http://www.macromedia.com/software/director/.
25. Moran, T.P. and Dourish, P., Introduction to This Special Issue on Context-Aware Computing. *Human-Computer Interaction*, 2001. 16: pp. 87-95.
26. Philipose, M., et al., Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing*, 2004. 3(4): pp. 50-57.
27. Reilly, D., et al., Evaluating Early Prototypes in Context: Trade-offs, Challenges, and Successes. *IEEE Pervasive Computing*, 2005. 4(4): pp. 42-50.
28. Rowan, J. and Mynatt, E.D., Digital Family Portrait Field Trial: Support for Aging in Place. In CHI'05, pp. 521-530.
29. POS Tagger, http://nlp.stanford.edu/software/tagger.shtml.
30. Suchman, L.A., *Plans and Situated Actions: The Problem of Human-Machine Communication*. 1987: Cambridge University Press.
31. Truong, K.N., et al., CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home In Ubicomp'04, pp. 143-160.
32. Weiser, M., The Computer for the 21st Century. *Scientific American*, 1991. 265(3): pp. 94-104.
33. WordNet, http://wordnet.princeton.edu/.
34. Xindice, http://xml.apache.org/xindice/.