Exploring Activity-Based Ubiquitous Computing: Interaction Styles, Models and Tool Support

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

[1]DUB Group
Computer Science and Engineering
University of Washington
Seattle, WA 98105-4615 USA
{yangli, landay}@cs.washington.edu

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

ABSTRACT
Activity-based computing is a promising paradigm for ubiquitous computing. By providing a consistent framework and structural view for integrating ubicomp technologies into natural human activities, activity-based computing can better facilitate our day-to-day lives and allow ubicomp technologies that are sustainable in the dynamic, complex world in which we live. In this position paper, we surface several aspects of activity-based computing. We first briefly describe what activity-based computing is. We discuss our early ideas on an activity framework for structuring ubicomp technologies and representing interaction contexts. Next, we discuss the interaction styles and models of this new paradigm. We then discuss tool support for activity-based computing. In particular, we give an overview of our ongoing work on analysis/design tools supporting early stage prototyping of activity-based ubiquitous computing.

Author Keywords
Ubiquitous computing, activity-based computing, activity theory, context-aware computing, interaction styles / models, implicit interactions, rapid prototyping tools, activity/situation/action modeling.

ACM Classification Keywords
H.5.2 [**User Interfaces**]: Interaction Styles, Prototyping, Theory and Methods; D.2.2 [**Design Tools and Techniques**]: User interfaces

INTRODUCTION
The rapid development of technologies, as predicted by Moore's law, has provided us with tremendous computing power. However, as we move away from the well-understood desktop environment, we face a complex and dynamic real world [15] and, currently, we are extremely limited in leveraging this enormous computing power to facilitate our day-to-day lives in this complex environment.

Although steady progress is being made in developing individual ubicomp technologies (such as new sensors, mobile computing devices or ambient displays) as well as in using these technologies to enhance human experiences (such as showing nearby restaurants based on the user's current location), we argue it is important to explore a new interaction paradigm for the era of ubiquitous computing [21]. We believe a new interaction paradigm is important for providing guidelines for shaping and accelerating the development of ubicomp.

Here we describe our ongoing work on exploring activity-based computing as a new interaction paradigm for ubiquitous computing. This work was motivated by activity theory, a conceptual tool for describing human activities that originated in Russian psychology [17]. Activity theory provides a theoretical background on the structure of human activities and what to look at while analyzing human activities. This work was also based on our experience in striving to find a uniform framework for context-aware computing [11-13].

ACTIVITY-BASED UBIQUITOUS COMPUTING
We intend to explore activity-based computing as a new interaction paradigm for ubiquitous computing. We first discuss how human activities are structured in light of sociological and psychological findings [17, 19].

In the physical world, a human activity (such as keeping fit) often lasts for a relatively long period involving many discontiguous stages (such as running every morning for keeping fit) and many different situations (such as jogging in a park or exercising in a gym). An *activity* evolves every time it is carried out in a particular *situation*. A situation involves a set of *tasks* or *actions* performed under certain *conditions*, e.g., locations, times or temperatures. For example, for keeping fit (activity), a person gets healthier or stronger every time the person exercises in a Gym (situation) that may include doing pushups (action) or running on a treadmill (action).

In the entire process, tools (e.g., a microwave, a chair or a lamp) are a key element and important resource at every level of performing an activity. To facilitate our everyday lives, ubicomp technologies, as computing tools, should be designed based on activities. This includes organizing various ubicomp technologies based on the structure of activities and enabling the context-awareness of ubicomp technologies, i.e., knowing what people are doing.

To achieve these goals, we are designing an activity framework for 1) integrating ubicomp technologies into natural human activities, and 2) representing interaction contexts. Our proposed framework will involve key concepts such as activities, situations, actions, roles, tools and status properties. Roles are an abstraction of involved people based on what they do in an activity. We here briefly discuss tools and status properties.

For clarification and simplification, we use the term *toolets* for activity-based ubicomp tools (ubiquitous computing services as well as the hardware they are based on, such as sensors, mobile phones or ambient displays.) Computationally speaking, they are often at a smaller granularity than traditional applications are. Similar to determining graphical contexts in a scenegraph, the behavior of toolets can be contextualized based on how they are attached to an activity framework. For example, toolets attached to an activity node are designed for providing sustainable support for the lifetime of an activity (such as keeping track of how many steps a person has walked), while ones attached to a situation node address a particular situation of an activity (such as reminding me to take stairs rather than taking an elevator to improve fitness), which are only invoked in that situation.

Status properties characterize the aspects of interest or progress of an evolving activity. In the example of Elder Care, the activity is to help an elder age independently which can be characterized as a set of attributes such as an elder's health or medication status (e.g., blood pressure) and social involvement (e.g., the number of conversations). Status properties should be tracked throughout the lifetime of an activity and the history of status property changes is also important contexts for toolets to adapt their behaviors.

Toolets can be designed for facilitating an elder's daily routines (actions), such as prompting the elder to take medicines after meal or showing procedural instructions while making French toast (e.g., using an ambient display). Toolets can also be designed to capture the performance of these routines as well as the elder's status (e.g., using a sensor-embedded watch to record blood pressure or temperatures). In addition, toolets can be designed to communicate status properties to other roles in a care net, e.g., automatically showing the elder's status to family or more complex information to doctors.

Interaction Styles

Activity-based ubiquitous computing affords an *implicit* interaction style. Interactions in activity-based computing are mainly based on input such as what people are doing as well as in what conditions (e.g., locations and times) an activity is conducted. These inputs are acquired *implicitly* (e.g., via sensors). This includes not only what a person is doing at the moment but also what people have done in the past and if possible, what people plan to do in the future.

By leveraging implicit input, computer systems can decrease the requirement for user attention and enable users to focus on their tasks at hand. For example, computer systems can automatically keep track of an elder's daily activities and communicate her wellbeing to caregivers. Traditionally, a caregiver has to manually record an elder's activities using an ADL form.

At the same time, activity-based computing does not exclude explicit or traditional interactions. Instead, explicit and implicit interactions can work together as we previously explored for location-enhanced applications [12].

However, it is still an important research topic to further discover and design appropriate interaction styles for activity-based computing. What is the *interaction vocabulary* for users to explicitly communicate with a system in such a paradigm, e.g., allowing human users to explicitly correct inference errors? One important direction to explore is to design interactions based on existing skills, e.g., employing common hand gestures or vocal variations.

Interaction Models

One can use the well-known model of Model-View-Controller (MVC) to analyze the structure of activity-based computing. In MVC, views and controllers are provided for modifying application semantics, i.e., the model. A model may have multiple pairs of views and controllers that a user directly interacts with. It is interesting to notice that, in activity-based computing, an activity is the counterpart of a model in MVC and it can be realized and modified via situations that correspond views and controllers in MVC. In addition, actions decide how an activity should be performed in a particular situation, which is similar to how interaction is specified in a controller.

However, there is a significant distinction from the traditional use of MVC. In activity-based computing, how a user may act is mainly based on what people naturally do in the real world, which is more dynamic than what is predefined in a controller of a traditional UI. For example, interactions are performed via physical actions (with or without an object) under physical or social constraints in the real world rather than by manipulating electronic artifacts such as windows, menus and buttons on a computer screen. This implies the importance of analyzing and modeling activities, situations and actions so as to understand current practice and establish models that allow computer systems to automatically infer them for implicit interactions.

At a lower level, interaction sequences are traditionally modeled by state transition diagrams. For example, Jacob [8] designed a framework for using a set of coroutines of state transition diagrams to model multi-thread direct manipulation interfaces. Hudson [7] extended state transition diagrams with probabilistic reasoning for dealing with ambiguous interactions. Jacob et al. [9] enhanced discrete state machines for continuous interaction. Interestingly, the AI community has been using HMMs, also a state machine, to infer physical actions [18]. We are

exploring how these state-machine schemas can be combined or extended for modeling interactions based on real world actions and how they can fit into the overall structure of activity-based computing.

TOOL SUPPORT

In this section, we first survey what the field has achieved in tool support for activity-based computing. We then describe our ongoing work on analysis/design tools supporting early stage prototyping of activity-based ubiquitous computing.

To infer activities, Canny previously designed algorithms for factorizing activities from computer (or desktop) event logs [3]. Philipose et al. developed algorithms for inferring physical actions [18] based on user interactions with physical objects. Fogarty et al.'s work on interruptibility modeling can potentially support general situation modeling [5]. Bardram et al. [1, 2] has developed an activity-based computing infrastructure for managing mobile computing services, in which "activities" are a collection of computational services[1]. However, none of the work provides tool support for analyzing and designing activity-based computing.

Prior ubicomp prototyping tools (e.g., [12, 14]) addressed ubicomp design at a task or action level, such as helping me find a nearby library. Without enabling a higher level unit, namely activities, for design and analysis, it is hard to identify the crucial resources (such as roles and tools) that are constantly required across situations for carrying out an activity. It is insufficient to model and infer a richer runtime context, e.g., finding a nearby library for meeting a friend or learning astronomy. In addition, it is hard to organically integrate various ubicomp technologies to provide consistent support for the entire range of an activity.

Previously, activity-based design approaches [6, 17] have been proposed at a conceptual level, which has shown promise in areas such as CSCW, education and software engineering. However, these approaches are conceptual tools with abstractly defined principles and heuristics. In particular, activity theory, as the underlying component of these approaches, though conceptually powerful, is complex to understand and use in practice by general researchers and interaction designers. In addition, many technical challenges have to be overcome before conceptual analysis results can be realized as a testable ubicomp prototype.

Tools for Prototyping Activity-Based Ubicomp

We are currently exploring tool support for designing and prototyping activity-based ubiquitous computing, by which we intend to address several key challenges:

- Human activities are complex and dynamic. It is hard to *analyze activities* so as to identify the breakdowns of current practice that could be enhanced by ubicomp technologies (e.g., informing me of how grandma is doing in an unobtrusive way while I am working).
- It is often unclear what information is relevant for *modeling activities* and how much automatic sensing can do for us (e.g., is it relevant to telling whether this is a meeting by knowing whether Bob is in the room).
- Activity-based computing often employs *sophisticated technologies* (such as using accelerometers to infer whether a person is walking up stairs or taking an elevator) that require sophisticated technical expertise to build.
- It is hard to test and deploy activity-based ubicomp applications. To get valuable feedback, they should be *evaluated in-situ* and over a long period (how well a CareNet display can help family members to monitor the grandma's status without requiring them paying much attention in their daily lives [4]).

We have concretized activity-based design in the domain of ubicomp and devised an intuitive *Theme/Scene/Action* design paradigm to streamline the activity-based design process. The design paradigm allows researchers and designers to design activity-based computing using familiar concepts like those used in theaters [10]. A *theme* represents an activity that is conducted by human users (e.g., doing more exercise). A theme may be conducted in multiple *scenes* (e.g., running or riding a bike in a park). A scene represents a situation and a set of *actions* can be carried out in each scene. This framework allows designers to simultaneously analyze a target problem top-down and bottom up. A designer can first list themes involved in a problem and then enumerate concrete scenes where those themes are carried out. A designer can also start from concrete scenes of interest and then extract themes from scenes. The framework allows designers to establish a consistent structure over individual situations and independent ubicomp technologies. It will inform what individual toolets should be designed and how they should work together as a whole.

Based on this paradigm, we have built a tool called ActivityStudio that allows designers to design and analyze based on field data that reflects existing activities, model actions of interest using a video editing metaphor, prototype interactive behaviors of toolets by storyboarding and test a suite of toolets using a Wizard of Oz approach or real sensors.

To facilitate the design based on large scale field data, we plan to provide support for automatically extracting themes, scenes, and actions from collected log data such as data from in-situ studies. We also intend to allow analysis results as well as interaction designs to be shared and reused across problems and designers.

[1] In contrast, we intend to establish an interaction paradigm that integrates real world human activities and ubicomp technologies. In addition, we separate the notions of "activities" and "tasks" (or "actions") in light of activity theory. They are interchangeable in Bardram's framework.

Currently, a toolet can be rapidly prototyped using a built-in storyboard editor. However, we intend to enable the ActivityStudio to export analysis or modeling results so that a toolet can be prototyped or refined in other task or action-level prototyping tools (e.g., Topiary [12]). This will allow the ActivityStudio to leverage existing work that may have better support for particular domains or tasks.

In addition to making designs, our design environment could potentially be used for evaluating a design. From activity theory's point of view, the design and evaluation processes are all about analyzing human activities. The difference is before (for design) or after (for evaluation) ubicomp technologies have been designed and employed in human activities. We intend to build runtime support for collecting users' activity and interaction logs. This includes data collected by implicitly logging what users are doing or explicitly surveying users for their in-situ feedback. The collected data will then be analyzed in the environment for another iteration of a design. This supports a full cycle of an iterative design process.

Target Application Domains

To evaluate our framework as well as prototyping tool, we are currently working on two application problems: elder care and personal fitness. Some aspects of these problems have been studied in previous work [4, 16, 18, 20]. We intend to exercise our framework as well as prototyping tool on these problems. We hope to find out whether our approach can save effort in building such applications and also add new values.

We intend to build ubicomp support for the fitness domain, more concretely, to help people get more exercise and reach their fitness goals. By automatically keeping track of and visualizing the exercise that a user has done throughout a day, users can gain better self-awareness of their fitness progress. We also intend to encourage a user to do more exercise by providing hints appropriate to particular contexts (such as suggesting parking in a parking lot distant to their workplace).

We are also building ubicomp technologies to support elder care, e.g., helping elders age gracefully and independently. This activity is carried out in many situations such as in hospitals for medical attention, grocery stores for buying food and homes for dinning. There are multiple roles (such as elders, nurses, families, and friends) involved in a "care network". Ubicomp technologies, for example, can be built for detecting an elder's daily activities and communicating her status to caregivers in an unobtrusive way.

CONCLUSIONS

In this position paper, we explore activity-based computing as an interaction paradigm for ubiquitous/context-aware computing. We cover several issues of activity-based ubiquitous computing. We described our early ideas on an activity framework for structuring ubicomp technologies and representing interaction contexts. We discussed interaction styles, interaction models and tool support for activity-based ubicomp. We also briefly discussed our focus on providing tool support for early stage prototyping of activity-based ubiquitous computing.

Activity-based computing as proposed in this paper does not exclude previous work. Instead, we intend to provide a higher level framework for structuring ubiquitous and context-aware computing, where individual ubicomp technologies can come into play.

ACKNOWLEDGMENTS

We would like to thank John Canny, Tye Rattenbury and Bonnie Nardi for discussions on activity theory. We would also like to thank Beverly Harrison, Mike Chen and Sunny Consolvo for their feedback. This work was supported by Intel Research Seattle and the NSF under Grant No. IIS-0205644.

REFERENCES

1. Bardram, J.E., Activity-Based Computing: Support for Mobility and Collaboration in Ubiquitous Computing. *Personal and Ubiquitous Computing*, 2005. **9**(5): pp. 312-322.
2. Bardram, J.E. and Christensen, H.B. Open Issues in Activity-Based and Task-Level Computing. In Pervasive'04 Workshop on Computer Support for Human Tasks and Activities. 2004. Vienna, Austria.
3. Canny, J. GAP: A Factor Model for Discrete Data. In SIGIR'04. 2004. Sheffield, UK. pp. 122-129.
4. Consolvo, S., et al., Technology for Care Networks of Elders. *IEEE Pervasive Computing Mobile and Ubiquitous Systems: Successful Aging*, 2004. **3**(2): pp. 22-29.
5. Fogarty, J., et al., Predicting Human Interruptibility with Sensors. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2005. **12**(1): pp. 119-146.
6. 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.
7. Hudson, S. and Newell, G. Probabilistic State Machines: Dialog Management for Inputs with Uncertainty. In Proceedings of the ACM Symposium on User Interface Software and Technology. 1992. pp. 199-208.
8. Jacob, R.J.K., A Specification Language for Direct Manipulation User Interfaces. *ACM Transactions on Graphics (TOG)*, 1986. **5**(4): pp. 283-317.
9. Jacob, R.J.K., Deligiannidis, L., and Morrison, S., A Software Model and Specification Language for Non-WIMP User Interfaces. *TOCHI*, 1999. **6**(1): pp. 1-46.
10. Laurel, B., *Computers as Theatre*. 1993: Addison-Wesley Professional.
11. Li, Y., Hong, J.I., and Landay, J.A. ContextMap: Modeling Scenes of the Real World for Context-Aware Computing. In Adjunct Proceedings of UbiComp 2003: Poster. 2003. Seattle, WA. pp. 187-188.
12. Li, Y., Hong, J.I., and Landay, J.A. Topiary: A Tool for Prototyping Location-Enhanced Applications. In CHI Letters: 6(2), UIST'04. 2004. Santa Fe, NM. pp. 217-226.
13. Li, Y. and Landay, J.A. Rapid Prototyping Tools for Context-Aware Applications. In CHI'05 workshop - The Future of User Interface Design Tools. 2005. Portland, Oregon.

14. MacIntyre, B., et al. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In CHI Letters: 6(2), UIST'04. 2004. Santa Fe, NM. pp. 197-206.

15. Moran, T.P. and Dourish, P., Introduction to This Special Issue on Context-Aware Computing. *Human-Computer Interaction*, 2001. **16**: pp. 87-95.

16. Mynatt, E.D., et al. Digital Family Portraits: Supporting Peace of Mind for Extended Family Members. In CHI Letters: 3(1), CHI'01. 2001. pp. 333-340.

17. Nardi, B.A., ed. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. 1995, MIT Press.

18. Philipose, M., et al., Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing*, 2004. **3**(4): pp. 50-57.

19. Suchman, L.A., *Plans and Situated Actions: The Problem of Human-Machine Communication*. 1987: Cambridge University Press.

20. UbiFIT. Improving fitness through mobile devices. http://dub.washington.edu/projects/ubifit/

21. Weiser, M., The Computer for the 21st Century. *Scientific American*, 1991. **265**(3): pp. 94-104.