The Five Commandments of Activity-aware Ubiquitous Computing Applications

Nasim Mahmud, Jo Vermeulen, Kris Luyten and Karin Coninx

Hasselt University – tUL – IBBT,
Expertise Centre for Digital Media,
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{nasim.mahmud, jo.vermeulen, kris.luyten, karin.coninx}@uhasselt.be

Abstract. Recent work demonstrates the potential for extracting patterns from users' behavior as detected by sensors. Since there is currently no generalized framework for reasoning about activity-aware applications, designers can only rely on the existing systems for guidance. However, these systems often use a custom, domain-specific definition of *activity pattern*. Consequently the guidelines designers can extract from individual systems are limited to the specific application domains of those applications. In this paper, we introduce five high-level guidelines or *commandments* for designing activity-aware applications. By considering the issues we outlined in this paper, designers will be able to avoid common mistakes inherent in designing activity-aware applications.

Introduction

In recent years, researchers have demonstrated the potential for extracting patterns from users' behavior by employing sensors [1-3]. There are various applications for detecting the user's activities. Systems such as FolderPredictor [4] and Magitti [5] offer suggestions to users to assist them in their current activity. Other work has used activity recognition to provide awareness of people's activities to improve collaboration [6, 7] or the feeling of connectedness within groups [8]. Furthermore, a recent trend is to employ sensors to predict the user's interruptibility, allowing computers to be more polite and interact with users in an unobtrusive way [9-11].

As there is currently no generalized framework for reasoning about activity-aware applications, designers can only rely on the existing systems for guidance. However, these systems often use a custom, domain-specific definition of activity pattern. For example, the Whereabouts clock focuses on the user's location to determine their general activity [8], while FolderPredictor only takes the user's desktop activity into account [4]. Consequently, the guidelines designers can extract from individual systems are limited to that system's specific application domain. Although there are existing, focused design frameworks that deal with background and foreground interaction [12]; employing sensors [13-15]; or allowing users to intervene when a system acts on their behalf [16, 17], it is hard for designers to come to a generalized

body of design knowledge for activity-aware systems by integrating each of these frameworks.

In this paper, we introduce five high-level guidelines or *commandments* that need to be addressed when designing activity-aware applications. The main contribution of this framework is that it allows designers to avoid common mistakes inherent in designing activity-aware applications, regardless of the targeted application domain and activity recognition technologies. We combine and generalize existing models in a convenient way for designers. Just as for designing desktop applications, designers need general guidelines that they can rely on. The availability of such a set of guidelines is a critical factor in moving activity-aware applications beyond research prototypes and into practical applications. Our hope is that this work is another step towards a generalized body of design knowledge for activity-aware systems.

Our own work on applications for detecting user's activities with sensors and a broad study of existing activity-aware systems and design frameworks led us to develop the following *five commandments for activity-aware systems*:

1. View activity patterns in context;
2. Don't view a user's activities in isolation, but in their social context;
3. Deal with hierarchical reuse of patterns;
4. Take uncertainty into account at different levels of abstraction;
5. Allow users to intervene and correct the system.

In the following, each commandment is explained into detail together with a motivating example.

View activity patterns in context

Human activity does not consist of isolated actions, it is rooted in context. As discussed by Suchman [18], people's behavior is contextualized, i.e. the situation is a very important factor in determining what people will do. It is not possible to generalize and predict people's behavior without considering the situation they are in at that time. It is important that designers consider activities in context. Context consists of many aspects, including but not limited to: the time of day, location or the presence of other people (see commandment 2).

In activity-aware systems, an important aspect is how the elements inside a pattern are temporally related to each other. The time of occurrence of an element in a pattern, its duration and the time interval between two such elements are important issues to consider. The elements might form a sequence, occur concurrently, or have a more complex temporal relationship with each other such as the ones described by Allen [19]. It is important to take these temporal relationships into account in order to correctly identify different activity patterns.

Another use for time as context information is in supporting continuous interaction, as described by Abowd et al. [20]. Although desktop computers allow for multi-tasking, computer systems still largely expect users to do their tasks on a single machine, and to do one task after another (although they can switch quickly by switching from one window to another). Abowd et al. state that this assumption will not be valid for ubiquitous computing. In real life, people also regularly start new

activities and stop doing others, and do activities concurrently (e.g. watching television while ironing clothes).

For a practical method of implementing these features, we refer to Patterson et al. [21] who evaluated several techniques to detect activities, and explain how to support detection of concurrent and interruptible activities.

Don't view a user's activities in isolation, but in their social context

As we discussed before in commandment 1, human activity is highly dependent on the context. An important aspect of context is social context. A user's social context consists of the people who are in his surroundings. A user's activity patterns might be different when he is alone than when he is in a group. What's more, they might differ according to who he is with at that time (e.g. his colleagues, his friends or his family). Suppose Jim has a pattern which consists of playing his favorite music when he gets home from work. This pattern might not be applicable to the situation where he comes home and there are guests in the house.

Besides the fact that social context is important to consider for classifying patterns, analyzing the patterns of groups of people might help to create common patterns across people. Someone might only once enter a seminar room and turn off the lights when a speaker is going to give a talk, making it impossible to detect this as a pattern. However, several people might perform this activity in a common pattern. On the other hand, some patterns are specific to certain people (e.g. taking extra strong black coffee from the vending machine). This pattern might not necessarily hold for this person's colleagues.

Allow hierarchical reuse of patterns

Activity Theory (AT) defines an *operation* as an atomic procedure. A set of operations forms an *action* while a set of actions forms an *activity* [22]. The level of granularity can be determined according to what the sensors allow the system to detect. For example, in a GUI environment *operations* could be keyboard and mouse events, an *action* could be "selection", while the activity could be "select a file".

A set of *actions* may be reoccurring and can form an *activity pattern*. Existing patterns might themselves be used again in another higher level pattern (e.g. *making coffee* could be a sub pattern in the *morning routine* pattern). While the terminology is not clearly defined (*making coffee* could become an action while it was previously an activity), the point we want to make here is that hierarchical patterns are a natural way of describing real-life activities. According to AT [22], interaction between human beings and the world is organized into subordinated hierarchical levels. Designers should make sure that activity patterns can be organized in a hierarchy.

Take uncertainty into account at different levels of abstraction

An activity-aware system should take uncertainty into account at different levels of abstraction. Activities are in general detected by aggregating the values of one or more sensors and reasoning about these values to detect patterns. Sensors can be pieces of hardware (e.g. a temperature sensor), software (e.g. the user's schedule obtained from their online calendar) or a combination of the two (e.g. a GPS beacon).

At the lowest level, uncertainty can occur due to imprecision in the data generated by sensors. A sensor could be faulty, or might need to be smoothed over time to get correct results. For example, the user's distance to a display could be detected with a Bluetooth beacon attached to the display. The sensor would retrieve the Received Signal Strength Indication (RSSI) value from the user's Bluetooth-enabled phone to the beacon. When the RSSI value is low (around zero) the user is standing close to the display, otherwise he will be further away. However, this value will fluctuate a lot, as shown in Figure 1 where the phone is just laying on a desk. The system should deal with this uncertainty by smoothening the sensor reads.

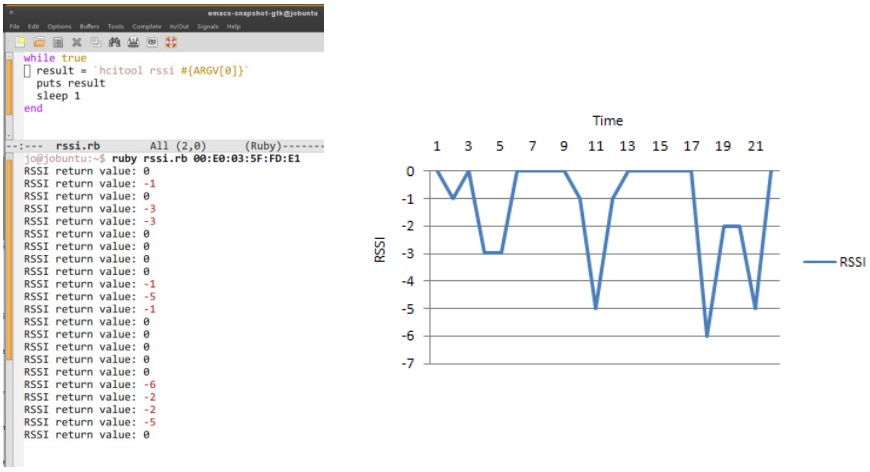

Fig. 1. Uncertainty at the lowest level: imprecision in data generated by a Bluetooth RSSI distance sensor.

At an intermediate level, there can be uncertainty in pattern recognition. This type of uncertainty can have several causes. It could be caused by inadequate sensors that prohibit certain parts of the user's state from being detected (e.g. the user's emotional state). Another cause could be insufficient training data for the recognition algorithm. For example, systems that are designed to be used in public spaces might not be able to learn a lot about their users, since most people will only use the system once. Uncertainty at an intermediate level can furthermore be caused by user actions that the designers of the system didn't take into account. As an example of this kind of uncertainty, consider an anti-carjacking device that will automatically be triggered when the driver exits the car with the engine still running. The system will then after a certain time automatically disable the car's engine, close the doors and sound an

alarm. The motivation for these actions is to make the car unusable when the driver has been unwillingly forced to exit the vehicle. Now suppose Jim is using his car to deliver a local community magazine in the neighborhood. At each house, he parks his car next to the road with the engine running, steps out to drop the magazine in the mailbox, and gets back in the car. When he gets to his friend Tom's house, he parks his car on Tom's driveway. After dropping the magazine in the mailbox of his friend Tom, he sees Tom working in the garden and goes over to chat with him. The engine of Jim's car is still running, but he feels his car is safe on Tom's driveway. Besides, they are both nearby. About a minute later however, the engine of Jim's car suddenly shuts down, the doors close and a loud alarm starts blaring. Jim is unable to enter his car and has to explain the situation to the authorities who arrive soon after. This embarrassing situation occurred because the designers of the anti-carjacking device did not take into account that a driver might ever step out of the car and leave his engine running for more than two minutes.

Finally, at the highest level, there might be uncertainty in the user's mental model of the system. This contributes to the discrepancy between what the user expects (their mental model of the system), what the system can sense through its sensors (e.g. a positioning system might not be as accurate as the user expects, leaving him wondering why his movements are left undetected), and what is desired (what is needed for the application), as discussed by Benford et al. [14]. They argue that by explicitly analyzing mismatches between each of *expected*, *sensed*, and *desired*, designers can detect problems resulting from these mismatches early on as well as find opportunities to exploit these mismatches.

An interesting practical, activity-aware application that deals with uncertainty is described by Patterson et al. [21]. Their system allows fine-grained activity recognition by analyzing which objects users are manipulating by means of Radio-Frequency Identification (RFID) tags attached to kitchen objects, and a glove equipped with an RFID reader that is worn by the user. Their system is resilient against intermediate level uncertainty: it can recognize activities even when different objects are used for it (e.g. using a table spoon instead of a cooking spoon for making oatmeal).

An important way to deal with uncertainty is to involve the user. The system could allow users to override its actions at any time, thereby offering a safety net for when the system's actions would be inappropriate. When a system is uncertain, it might also ask the user for confirmation. In an ideal case, computation would be split between humans and computers, letting each one perform the tasks they do best [23]. The next commandment discusses user intervention in activity-aware systems.

Allow users to intervene and correct the system

Users should be able to correct the activity-aware system when it makes a mistake. The system is bound to make mistakes, as it is impossible to detect the user's activity correctly in *every* possible situation. General activity recognition might be classified as an *AI-complete* problem [24], which means that a solution to this problem would presuppose the availability of a solution to the general problem of intelligence.

Since it is inevitable that the system will make mistakes, there should be a way for users to correct the system, so they can stay in control. As discussed by Bellotti et al. [13], the lack of communication between a system that employs sensors and its users is an important problem. Because these systems often have both less means of providing feedback and will take more actions on behalf of the user than desktop systems, the user will quickly feel out of control.

Dey et al. [17] introduce the term *mediation* to refer to the dialogue that takes place between user and system to handle ambiguities in context. They describe four guidelines for mediation of context-aware systems:

1. Applications should provide redundant mediation techniques to support more natural and smooth interactions;
2. Applications should facilitate providing input and output that are distributed both in space and time to support input and feedback for mobile users;
3. Interpretations of ambiguous context should have carefully chosen defaults to minimize user mediation, particularly when users are not directly interacting with a system;
4. Ambiguity should be retained until mediation is necessary for an application to proceed.

Guideline 1 deals with providing several redundant levels of interaction for user input and system feedback. This could for example range from most implicit to most explicit, depending on the user's attention and level of engagement in the task. Guideline 2 points out the fact that communication between system and user should take into account the space through which the user is moving, and have a timeout period after which a user might not have the chance to interact with a mediator anymore. Guidelines 3 and 4 refer to the fact that mediation should be used with care, so as not to distract the user unnecessarily.

To allow corrections to be made in a timely fashion, systems should make clear what they perceive of the user, what (automated) actions they are performing or going to perform and of course provide a way to undo or correct actions. Ju et al. [16] discuss three interaction techniques in their implicit interaction framework that cover these requirements: *user reflection* (making clear what the system perceives of the user); *system demonstration* (showing what actions the system is performing or going to perform); and *override* (providing a handle for the user to correct the system).

To illustrate the necessity of mediation, we discuss an experience of incorrect system behavior that one of the authors had when visiting a house of the future. This exhibit demonstrated how recent technology trends such as context-awareness could influence our life in the future, and might make our homes *smart*. The author missed his train and arrived a bit too late. He had to enter the seminar room when the first talk had already started, which was very annoying since the only entrance to the room was in front of the room, next to the lecturer. As if this wasn't enough, the *smart* room automatically turned on the lights when he entered the room, leaving no escape to an unremarkable entrance.

This experience clearly illustrates that activity-ware systems need mediation to ensure that users remain in control and will not get frustrated. If we would apply the three interaction techniques of Ju et al. [16] to this scenario, the system might indicate that it senses that the user is entering the seminar room (*user reflection*), announce

that it is going to turn on the lights (*system demonstration*), and – most importantly – provide the user with an opportunity to cancel this action (*override*). The authors believe that designers of activity-aware systems should always keep user intervention in mind. Both the guidelines for mediation [17] and the implicit interaction techniques [16] are useful to take into account for this purpose.

Conclusion

In this work we have presented *five commandments* of activity-aware ubiquitous computing applications which is a design guideline for application designers. The commandments presented in this paper is high level guideline intended to help designers in designing activity-aware systems. The guideline will help designers to avoid common mistakes in designing activity-aware systems.

Existing activity-aware applications often use a domain-specific definition of activity pattern. Thus existing works do not provide a generalized body of knowledge that application designers can rely on. This work is a step towards generalized guidelines for the application designer. The commandments in this paper suggest the designer to consider the users' activity in their social context; to consider a pattern of activity in context and to allow for hierarchical reuse of patterns. The commandments also suggest the designer to consider uncertainty at different levels of abstraction and to allow users to intervene and correct the system when the system makes mistake.

For designing desktop applications, there are guidelines allowing the designers to use off-the-shelf knowledge. This is not the case for activity-aware ubiquitous computing applications. Hence, generalized guidelines for designing these applications are necessary to move them from lab prototypes into commercial development.

In our ongoing work, we are developing an activity-aware ubiquitous computing system that takes the five commandments into account. In our future work, we want to ensure the validity of our guideline in practical settings.

Acknowledgements

Part of the research at EDM is funded by ERDF (European Regional Development Fund) and the Flemish Government. Funding for this research was also provided by the Research Foundation -- Flanders (F.W.O. Vlaanderen, project CoLaSUE, number G.0439.08N).

References

1. Begole, J.B., Tang, J.C., Hill, R.: Rhythm modeling, visualizations and applications. Proceedings of the 16th annual ACM symposium on User interface software and technology. ACM, Vancouver, Canada (2003)

2. Eagle, N., Pentland, A.: Reality mining: sensing complex social systems. Personal Ubiquitous Comput. **10** (2006) 255-268
3. Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hahnel, D.: Inferring Activities from Interactions with Objects. IEEE Pervasive Computing **3** (2004) 50-57
4. Bao, X., Herlocker, J.L., Dietterich, T.G.: Fewer clicks and less frustration: reducing the cost of reaching the right folder. Proceedings of the 11th international conference on Intelligent user interfaces. ACM, Sydney, Australia (2006)
5. Bellotti, V., Begole, B., Chi, E.H., Ducheneaut, N., Fang, J., Isaacs, E., King, T., Newman, M.W., Partridge, K., Price, B., Rasmussen, P., Roberts, M., Schiano, D.J., Walendowski, A.: Activity-based serendipitous recommendations with the Magitti mobile leisure guide. Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. ACM, Florence, Italy (2008)
6. Tullio, J., Goecks, J., Mynatt, E.D., Nguyen, D.H.: Augmenting shared personal calendars. Proceedings of the 15th annual ACM symposium on User interface software and technology. ACM, Paris, France (2002)
7. Isaacs, E.A., Tang, J.C., Morris, T.: Piazza: a desktop environment supporting impromptu and planned interactions. Proceedings of the 1996 ACM conference on Computer supported cooperative work. ACM, Boston, Massachusetts, United States (1996)
8. Brown, B., Taylor, A.S., Izadi, S., Sellen, A., Kaye, J.J., Eardley, R.: Locating Family Values: A Field Trial of the Whereabouts Clock. Lecture Notes in Computer Science **4717** (2007) 354
9. Gibbs, W.W.: Considerate Computing. Scientific American **292** (2005) 54-61
10. Fogarty, J., Hudson, S.E., Atkeson, C.G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C., Yang, J.: Predicting human interruptibility with sensors. ACM Trans. Comput.-Hum. Interact. **12** (2005) 119-146
11. Horvitz, E., Koch, P., Apacible, J.: BusyBody: creating and fielding personalized models of the cost of interruption. Proceedings of the 2004 ACM conference on Computer supported cooperative work. ACM, Chicago, Illinois, USA (2004)
12. Buxton, B.: Integrating the Periphery and Context: A New Taxonomy of Telematics. Proceedings of Graphics Interface '95 (1995)
13. Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A., Lopes, C.: Making sense of sensing systems: five questions for designers and researchers. Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. ACM, Minneapolis, Minnesota, USA (2002)
14. Benford, S., Schnädelbach, H., Koleva, B., Anastasi, R., Greenhalgh, C., Rodden, T., Green, J., Ghali, A., Pridmore, T., Gaver, B., Boucher, A., Walker, B., Pennington, S., Schmidt, A., Gellersen, H., Steed, A.: Expected, sensed, and desired: A framework for designing sensing-based interaction. ACM Trans. Comput.-Hum. Interact. **12** (2005) 3-30
15. Hinckley, K., Pierce, J., Horvitz, E., Sinclair, M.: Foreground and background interaction with sensor-enhanced mobile devices. ACM Trans. Comput.-Hum. Interact. **12** (2005) 31-52

16. Ju, W., Lee, B.A., Klemmer, S.R.: Range: exploring implicit interaction through electronic whiteboard design. Proceedings of the ACM 2008 conference on Computer supported cooperative work. ACM, San Diego, CA, USA (2008)
17. Dey, A.K., Mankoff, J.: Designing mediation for context-aware applications. ACM Trans. Comput.-Hum. Interact. **12** (2005) 53-80
18. Suchman, L.: Plans and Situated Actions: The Problem of Human-Machine Communication. Cambridge University Press (1987)
19. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26** (1983) 832-843
20. Abowd, G.D., Mynatt, E.D.: Charting past, present, and future research in ubiquitous computing. ACM Trans. Comput.-Hum. Interact. **7** (2000) 29-58
21. Patterson, D.J., Fox, D., Kautz, H., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on (2005) 44-51
22. Leontiev, A.N.: Activity, Consciousness and Personality. Englewood Cliffs, NJ: Prentice-Hall
23. Horvitz, E.: Reflections on Challenges and Promises of Mixed-Initiative Interaction. AI MAGAZINE **28** (2007) 19
24. Mallery, J.C.: Thinking about foreign policy: Finding an appropriate role for artificial intelligence computers. The 1988 Annual Meeting of the International Studies Association, St. Louis, MO