Towards Automatic Device Configuration in Smart Environments

Kay Connelly and Ashraf Khalil

Indiana University Computer Science Department
150 S Woodlawn Ave.
Bloomington, IN 47405-7104
{connelly, akhalil}@cs.indiana.edu

Abstract. As the number of mobile devices we carry grows, the job of managing those devices throughout the day becomes cumbersome. In addition, the ability to use mobile devices in unethical ways without being detected is becoming widespread [13-14]. It is desirable for mobile devices to automatically configure themselves based on the context of the environment and user preferences for both convenience and security purposes. Having the cell phone switch the ringer off and turn on the vibrate notification in a movie theatre is convenient to the user; while having the video phone disable its video capabilities in a locker room is necessary for others' privacy. We present a preliminary architecture for automatic device configuration in smart environments.

1 Introduction

Mobile and wearable computational devices are becoming widespread. Beyond laptops and PDAs, intelligent devices such as watches, active badges, cell phones and mp3 players are beginning to interact with our environments as part of our everyday lives. For example, IBM's Linux watch can be used as an authentication device [15], whereas the active badge can be used to locate resources close to the badge wearer [12].

How we use such devices often changes depending on our social context. For example, a user may want their mp3 player to turn off once they enter an office, or their cell phone ringer to be disabled during an important meeting. As the number of devices we carry with us grows, managing such devices throughout the day as our context changes can become overwhelming.

In order to make device management less cumbersome, and thus the devices more usable, devices should be able to automatically reconfigure themselves based on the current context and user preferences. In addition, as abuse of ubiquitous computing devices becomes more prevalent [13-14], systems that can in some sense force configuration changes on devices become interesting. For example, following legislation that outlaws the use of video phones in locker rooms, a gym may want the

ability to temporarily turn off the camera capabilities of a video phone. Similarly, a university may wish to disable all cell phone calls (except for an outgoing 911) in a lecture hall when an exam is scheduled.

In this paper, we describe a preliminary architecture for automatic device configuration. In section 2, we review the related work. In section 3, we give a detailed motivating example based on cell phone usage. The design of our system is presented in Section 4. We discuss the open research issues in Section 5 and conclude in Section 6.

2 Related Work

Our work is within the framework of a *smart space*. There are many different terms used in the literature for such an environment. Smart home, smart office and interactive workspace are examples of the names that refer to a physical space that is augmented with smart devices and applications to support an interactive environment for the users that help them to achieve their tasks efficiently. Some smart space projects that satisfy this definition are Gaia [6], i-LAND [7], Oxygen [5], Aura [4], EasyLiving [2], and Interactive Workspaces [1,3].

Both Gaia and Interactive Workspaces support automatic integration of devices in the smart, interactive space. Mobile devices automatically register themselves within the space. Most smart environments focus on integrating the services offered by mobile devices with the space services and vice versa. For example, Gaia's active space is equipped with infrared beacons that broadcast information about the space. A smart device uses that information to broadcast heartbeat events to the space. The space then contacts the device to retrieve the description of that device. Although smart environments offer automatic integration of the devices, they do not try to enforce certain usage policies or to agree on what is allowed and not allowed within that space.

Other work in the field of the interaction between devices and the environment around them focuses on empowering a certain type of device to be more environment-aware in order to adapt their behavior accordingly. An example of this is the SenSay project [9] where the cell phone is augmented with sensors and other sources of information to sense the context of the user and adapt the cell phone functionality accordingly. This work focuses on user convenience without taking into account the environment policy. Similar works that focus on specific classes of devices without taking into consideration the policy of the surrounding environment can be found in [10, 11].

3 Motivating Example

We use cell phones as our motivating example. Based on social context, the user may want their cell phone to behave in different ways. For example, when a movie starts

in a theatre, a user may want their ringer to be disabled and the vibrate notification turned on. Once they leave the theatre, they would like the phone to return to its normal configuration with the ringer enabled. Alternatively, the user may not wish to be disturbed at all and instead have the cell phone take messages and notify the user of the messages once the movie is over. With the combination of caller ID, a user's configuration preferences can become arbitrarily complex based on the importance of the caller to the user.

Aside from user preferences, the smart space environment may have its own policies when it comes to devices brought into that space. The theatre, for example, could have a policy that cell phone ringers are not allowed to ring, as that would disturb their other customers. Determining when a space owner's wishes override an individual user's preferences is an interesting societal question. However, there are certainly some cases which will be driven by law. As abuses such as the unauthorized taking of pictures of undressed patrons in locker rooms become prevalent, the law will eventually catch up to the technology and such practices will be outlawed. Mechanisms to enforce such space-driven configurations need to be developed.

4 Design

We make two main assumptions in our design. First, the smart space environment can infer the social context. It is not unrealistic for us to separate the contextual evaluation from the context-driven application. Indeed, Salber et al. discusses the need for this separation in order to ease the development of context-aware applications [8].

Our second assumption is that devices, or more specifically applications running on devices, can be standardized. For example, all cell phones have similar functionality: incoming calls, outgoing calls, standard mechanisms to notify users of incoming calls, et cetera. Standardizing how applications describe that functionality will aid in the deployment of our automatic configuration software. Even once cell phones are no longer single-use devices but are integrated with other devices (mp3 player, digital camera, etc…), the cell phone applications still have the same functionality and can be described in a standard way.

Figure 1 gives a picture of our architecture. In our architecture, there are two basic entities: devices[1] and spaces. There are 4 phases of interactions. Phase 1 is the initialization where the device and space are configured by their owners. Phase 2 is the registration where the space and device policies are first examined. Phase 3 is the policy resolution, where the device and space owners are consulted if there is a conflict in the space and device policies. Phase 4 is when the device is configured as agreed upon.

[1] With multimodal devices, we would consider a particular application as an entity instead of a device.

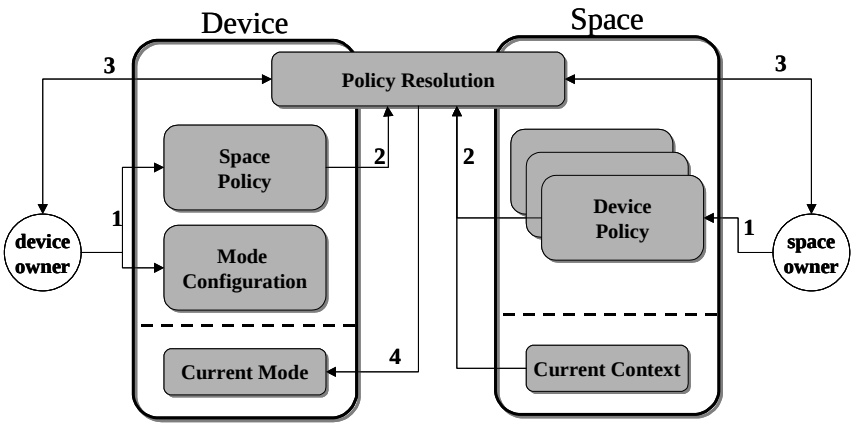

Figure 1: Architecture for automatic device configuration in smart space environments.

Devices

Each device type has a standard set of *modes*, which indicates how the device should behave based on the context. For our cell phone example, there might be the following modes:

- **Quiet Mode:** the ringer should be disabled
- **Noisy Mode:** the user is in a loud environment
- **Disable Mode:** all incoming and outgoing calls should be disabled
- **Default Mode:** normal conditions

Other types of devices will have modes that are specific to their functionality. By standardizing the set of modes for a particular type of device, a smart space need only install that type of device once, instead of installing every model of a particular device[2].

Each mode can be configured by the user according to limited choices provided by the manufacturer. The manufacturer options should satisfy the mode's constraints. The quiet mode, for example, could have any incoming call notification mechanism *except* for the ringer. So, the user could choose:

- vibrate notification
- flash lights on keypad/screen notification
- take a message and notify later
- play a message telling caller to call back later

[2] In reality, devices constantly gain new features leading to different versions of the standardized sets of modes for new generations of devices. Each new version will have to be installed in a space.

The particular configuration that a user chooses for all of the modes is called a device's ***mode configuration*** and is based purely on users choosing an available configuration for each mode.

The device also has a device's ***space policy***, which indicates how a particular space can change the current mode. For example, a user may not want a space to be able to put their phone into disable mode (at least, not without their explicit consent), or a user may completely trust a space they personally manage, such as their home, but want notification whenever public spaces such as a restaurant tries to change their device's mode.

Both the mode configuration and the space policy are determined by the user during phase 1 of Figure 1.

Spaces

A space maps a social context to device modes. The mapping is called a space's ***device policy***. For example, the administrator of a theatre may specify that when a movie is playing, a cell phone device should be in quiet mode. A lecture hall administrator may specify that cell phones be in quiet mode during a lecture and in disable mode during an exam. The device policy is determined by the space owner during phase 1 of Figure 1. While we use the context information in our architecture, we do not address how the space infers this context.

When a device enters a space, it registers with that space. The first time a user enters a space with a device, the device's space policy and the space's device policy must be examined for conflicts, as corresponds to phases 2 in Figure 1. If there are no conflicts, the owners need not be consulted, skipping immediately to phase 4 where the device mode is set. If, however, there is a conflict, the conflict must be resolved by interacting with the device and/or space owner, as shown in phase 3 of the figure.

Returning to our example, a user may specify that no space is allowed to put their phone into disable mode. At the same time, a lecture hall specifies that all phones should be disabled during an exam. In such a situation, either the user or the space must relent and this requires user interaction. Initially, the device can interrupt the user and describe the conflict, attempting to get the user to resolve the problem. The user could decide to accept the space's policy always, this one time, or not at all. If the user rejects the space policy, there must be a mechanism to notify the space owner (or current owner, in this case, the instructor), that a device is not configured properly[3]. Then, it is up to the user and the owner to resolve the problem. Knowing that a particular student's cell phone is not disabled may allow the instructor to monitor that student during the exam more carefully, therefore not requiring the conflict to be resolved in the virtual world.

[3] Of course, if a space owner is not available, there should be a mechanism to at the very least log the information regarding the conflict for future analysis by the space owner.

Policy Resolution

There are three major issues with policy resolution: *when* the conflicts should be resolved, *how* they should be resolved and *where* the policy resolution logic resides.

For complex policies, many social contexts may rarely be encountered. An exam, for example, is a relatively rare instance in a lecture hall. Should a potential conflict within the context of an exam be brought to the attention of a student every time she attends class? If it is, wouldn't the student eventually accept the space policy out of sheer annoyance, then forget the implications later when the context is actually valid?

Alternatively, if a conflict is resolved only when it is relevant, and if a space changes context frequently, the device may frequently interrupt its owner to resolve configuration problems. Such a high annoyance factor could lead the owner to adopt a lax space policy.

With both of these cases, there runs a risk that the device is automatically configured in ways the owner does not anticipate or want. For automatic configuration to become viable, a balance must be achieved that makes policy resolution manageable.

In terms of how conflicts should be resolved, our initial policy resolution algorithm brings every policy conflict to the attention of the device owner, and possibly to the attention of the space owner. This approach, however, can become unmanageable as device and space owners attempt to further restrict the actions of the other. While relaxing the strict adherence to the policies can allow policy resolution to occur without user intervention, it can also lead to abuse by either the device or space owner and confusion about how the configuration is determined. Our architecture allows for experimentation with different policy resolution techniques in order to minimize user interaction *and* confusion.

The third concern is where the policy resolution logic resides. There are security concerns about a device or space allowing access to their policies by untrusted entities. Our architecture does not yet address these concerns, but acknowledges that both entities will have to be involved in the resolution strategy in some way.

5 Discussion and Future Work

While we have a prototype implementation of our design, there are several open issues which we have yet to address, ranging from user acceptability to security to enforcement. Our current implementation uses XML to describe the space and device policies. The policy resolution occurs at device registration time, with the policy resolution logic contained in the space service.

As discussed in Section 4, our architecture does not mandate when, how or where the policy resolution must occur. We feel that this ambiguity is necessary for the

preliminary stages of our work. Indeed, determining the answers to when, how and where are the next major steps for our project.

For the question of when, we intend to combine techniques in usability analysis to evaluate several possible implementations in an effort to find a balance that makes automatic device configuration viable.

For the question of how, we are investigating multi-agent negotiation techniques in order to minimize user distraction. By treating both device and space owner policies as negotiable, it may be possible to use game theory techniques in designing the policy resolution protocols in order to maximize specific criteria [16]. Further, different criteria could have different priorities. For example, legal use is always enforced, followed by maintaining security of devices and spaces, followed by achieving acceptable, if not ideal, configuration for participating parties.[4]

For the question of where, we will evaluate the security implications of making configuration policies public, as well as look at techniques in the mobile agent research which makes different assumptions on the trustworthiness of hosts and agents.

As the number of devices and the number of contexts grow, it is likely that a device will not behave as expected by the user all of the time. A major cause of unintended side effects is that the device owner and space owner will most often be different people with different goals and models of use. It is important that we recognize the likelihood of unanticipated behavior; and while we will strive to reduce their frequency, we must develop mechanisms for users to determine the reason a device is configured a certain way and the ability to make corrective actions for the future.

While our architecture can easily suggest configuration modes to a device, the issue of enforcement is still open. Certain devices from manufacturers can be designated as "compliant". But, just as banning video phones won't get rid of regular cameras, adopting our infrastructure will not rid the world of non-compliant video phones. And while the cell phone manufacturers could adopt an industry-wide standard, there are many types of devices that are programmable, such as PDAs, for which a simple compliance tag will not suffice. Ultimately, a device owner can program certain devices to appear to be compliant, when in reality, they are not.

---

[4] This approach is somewhat reminiscent of the Three Laws of Robotics as presented in Isaac Asimov's fictional work, *I, Robot*, where the prioritized "laws" are used to describe acceptable robot behavior. Within the three laws, never causing harm to humans supercedes never causing harm to self, which supercedes fulfilling requests from humans.

6 Concluding Remarks

We have presented a framework for automatic device configuration for smart spaces. This work was inspired by the recent advances in smart spaces and by the introduction of many new smart mobile devices and gadgets. By having automatic device configuration we can minimize user distraction and can get closer to achieving the goals of ubiquitous computing. We discussed four main phases of the prototype and three different main components. We are experimenting with different implementations of our automatic device configuration design. While we are using a cell phone scenario as our guiding example, our framework and implementation can be generalized for any mobile smart device.

Aside from our implementation efforts, we envision several areas for future work. These involve the problem of resolving the policy conflict with a minimum amount of user distraction and running usability testing to test our methods. In addition, we plan to address the critical issue of privacy as well as the issue of integrating our model with real devices.

Acknowledgments

This work was partially supported by a grant from the Lilly Endowment and the Center for Applied Cybersecurity Research at Indiana University.

References

1. A. Fox, B. Johanson, P. Hanrahan, and T. Winograd, "Integrating Information Appliances into an Interactive Workspace", IEEE Computer Graphics and Applications', May/June, 2000, pp. 54-65.
2. Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S., "EasyLiving: Technologies for Intelligent Environments", Handheld and Ubiquitous Computing, September 2000.
3. Brad Johanson, Armando Fox, Terry Winograd, "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms", IEEE Pervasive Computing Magazine 1(2), April-June 2002.
4. Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P, "Project Aura: Toward Distraction-Free Pervasive Computing", IEEE Pervasive Computing, April-June 2002.
5. MIT Project Oxygen, http://oxygen.lcs.mit.edu.
6. Manuel Román, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces", IEEE Pervasive Computing, pp.74-83, Oct-Dec 2002.

7. N. Streitz et al., “i-LAND: An interactive Landscape for Creativity and Innovation,” Proc. ACM Conf. Human Factors in Computing Systems (CHI 99), ACM Press, New York, 1999, pp. 120-127.

8. Salber, D., Dey, A.K., Abowd, G.D., “The Context Toolkit: Aiding the Development of Context-Enabled Applications”, in Proceeding of the CHI 99 Conference on Human factors in Computing Systems, Pittsburgh, Pennsylvania, United States, ACM Press New York, NY, USA (1999) 434--441.

9. Siewiorek, D., Smailagic A., Furukawa, J., Moraveji N., Reiger K., Shaffer J., “SenSay: A Context-Aware Mobile Phone”, Submitted to the International Symposium on Wearable Computers, 2003.

10. Schmidt, Albrecht, Antii Takaluoma and Jani Mntyjrvi, “Context-Aware Telephony over WAP”, Springer-Verlag, London, Ltd., Personal Technologies (2000), presented to Handheld and Ubiquitous Computing (HUC2k), September.

11. A. Schmidt and K.Van Laerhoven. “How to Build Smart Appliances?” IEEE Personal Communications, Aug. 2001, pp. 66 - 71.

12. Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons, “The Active Badge Location System”, In ACM Transactions on Information Systems, January 1992.

13. Jeffrey L. Popyack, Nira Herrmann, Paul Zoski, Bruce Char, Christopher D. Cera and Robert N. Lass . “Academic Dishonesty in a High-Tech Environment". Special Session at the 34th SIGCSE Technical Symposium on Computer Science Education. Reno, Nevada. February 19-23, 2003.

14. “Cam phones spread new brands of mischief.” CNN.com/Technology 10, July 2003: http://www.cnn.com/2003/TECH/ptech/07/10/naughty.camphones.ap/index.html

15. N.Kamijoh,T.Inoue, K.Kishimoto and K.Tamagawa. “Linux Watch: Hardware Platform for Wearable Computing Research”. In Second IEEE Pacific-RIM Conference on Multimedia, Beijing Oct 2001

16. Jeffrey S. Rosenschein and Gilad Zlotkin. “Rules of Encounter: Designing Conventions for Automated Negotiation among Computers”, MIT Press, Cambridge Massachusetts, 1994.