**AmIBusy: High-Level Abstraction in Ubiquitous Sensing Environments**

Position Paper for the
Pervasive 2004 Workshop on Toolkit Support for Interaction in the Physical World

James Fogarty
Human Computer Interaction Institute
Carnegie Mellon University
jfogarty@cs.cmu.edu

In a series of studies, we have examined the feasibility of creating sensor-based statistical models of the interruptibility of office workers [5], explored the degree of robustness that we might reasonably expect from such models [3], and studied human ability to use audio and video recordings to estimate the interruptibility of an office worker [2]. Our studies indicate that sensor-based statistical models can predict self-reports of interruptibility as well as or even better than people viewing audio and video recordings of office workers.

As a part of continuing this line of research, we are now building AmIBusy, a system for *Automatically Modeling Interruptibility By Unobtrusively Sensing You*. AmIBusy will use collected interruptibility observations, such as self-reports, and logs of sensor readings to automatically learn individualized sensor-based statistical models of interruptibility. In this position paper, I focus on AmIBusy as an example of a tool to provide application programmers with a high-level abstraction in a ubiquitous sensing environment.

As our world continues to be filled with many small computational devices, it is becoming increasingly unreasonable to expect that people will directly supervise, configure, or even bother to notice all of the computational devices in their lives. Programmers building applications for this world also cannot reasonably be expected to anticipate what devices might be present in an environment, but must instead be provided with useful abstractions.

Inspired by the abstractions provided in desktop user interface software toolkits [7], a variety of research projects are exploring abstractions of physical sensors and physical input devices. For example, Papier-Mâché can provide high-level events related to the appearance of an RFID near a reader, the scanning of a barcode, or a computer vision system, without requiring the application programmer to know exactly what model of RFID reader or what type of barcode is involved [6]. The Context Toolkit offers similar abstractions, such as informing an application that a person is present at a given location, without requiring the application to understand whether the notification was prompted by the detection of an RFID, the docking of an iButton, or the entry of a username/password combination [1].

AmIBusy seeks to provide a different type of abstraction. Rather than abstracting *how* information was collected, AmIBusy seeks to abstract *what* information was collected. To clarify, the deployment of many sensors and input devices into the world is currently hampered by the difficulty of deploying such devices and obtaining information from them. But once widely-deployed toolkits provide useful abstractions of devices like “object detector” (which detects the presence of some object), “person detector” (which is an object detector, but only detects people), and “named person detector” (which is a person detector that can report who has been detected), it seems that we may encounter a different problem. We might find that so much sensed information is available that application programmers will have a difficult time making

good decisions about what sensors are appropriate for an application. And even if an application programmer selects the best possible set of sensors, those sensors might not be available in all locations or new, more appropriate, sensors might become available shortly after the application programmer commits to the existing sensors.

AmIBusy seeks to mitigate these problems by providing a high-level abstraction that can be estimated from whatever sensors are available in an environment. Application programmers can use interruptibility in their programs (for example, delivering notifications only when a person is in their office and their interruptibility is medium or better [4]), and AmIBusy will automatically use interruptibility observations and the sensors available in an environment to estimate interruptibility. If new sensors become available or existing sensors are removed, AmIBusy will adapt its statistical models to provide the best model of interruptibility that it can create, without the need for additional attention from the application programmer.

While participating in this workshop, I hope to share the viewpoint that, although most systems currently use only a handful of sensors or physical input devices, we need to design our toolkits such that they might eventually support many devices. For example, a system that uses only a handful of sensors or physical input devices might subscribe to one by providing the toolkit with a unique identifier entered by the application programmer, but an automated system will instead want to query the toolkit for a list of available sensors and physical input devices. Similarly, an application programmer will probably have some knowledge of the location of the sensors and physical input devices they use, but an automated system may need to be able to query the toolkit for the location and other physical properties of devices. Note that this does not necessarily imply the tedium of maintaining a database containing information about each sensor. Instead, a toolkit might use its knowledge of how devices communicate to infer information about their relative location (a device communicating via short range radio is relatively close to the access point it is communicating with, and it is also relatively near any other devices communicating via the same access point). A variety of such issues will arise as we see the differences between an application programmer configuring a dozen sensors or input devices and an automated system examining the utility of hundreds of sensors and input devices.

**References**


1. Dey, A.K., Salber, D. and Abowd, G.D. (2001) A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal*, *16* (2-4). 97-166.
2. Fogarty, J., Hudson, S., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. and Yang, J. (2004) Predicting Human Interruptibility with Sensors. *To Appear, ACM Transactions on Computer-Human Interaction (TOCHI)*.
3. Fogarty, J., Hudson, S. and Lai, J. (2004) Examining the Robustness of Sensor-Based Statistical Models of Human Interruptibility. *To Appear, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2004)*.
4. Heer, J., Newberger, A., Beckmann, C. and Hong, J.I. liquid: Context-Aware Distributed Queries. In Dey, A.K., Schmidt, A. and McCarthy, J.F. eds. *"UbiComp 2003: Ubiquitous Computing" ACM UbiComp Proceedings*, Springer-Verlag, 2003, 140-148.
5. Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. and Yang, J. (2003) Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2003)*, 257-264.
6. Klemmer, S.R., Li, J., Lin, J. and Landay, J.A. (2004) Papier-Mâché: Toolkit Support for Tangible Input. *To Appear, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2004)*.
7. Myers, B., Hudson, S.E. and Pausch, R. (2000) Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *7* (1). 3-28.