Evolve in Heterogeneous Environments with Rich Components

Zebin Chen, Andrew Fortier, Craig Pataky, Xiangkui Yao, Terrence Waggoner, Amanda Hosler, Stephen Fickas

Wearable Computing Lab, University of Oregon, Eugene, OR 97403

Introduction

Recent advances in sensors, wireless communication and mobile computation provide low-cost circuit-units powerful enough to develop personal assistant devices.

But one manufacturer cannot make all needed hardware units. Furthermore, new circuit-units and new service requirements emerge all the time[1][2]!

A static approach doesn't work; wearable computing devices must evolve in heterogeneous environments

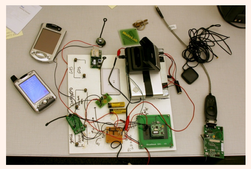


Fig. 1. A wearable computing device may interact with devices different in communication interface, power supply, reliability, etc.

The first step to GO!

In the Get Outside (GO) Navigation Project, our goal is to create new wearable computing devices and use-styles to enable brain injury survivors to venture out into the community.

As a part of the GO project, a travel bag system is built from a set of heterogeneous devices to help users make the first step out the door. The story is:

Fig. 2. (a) Furniture in a digital home. TV is used as a collaboration media for a brain injury patient, Bob[3]. **(b)** A care provider, Clark, inputs Bob's schedule.

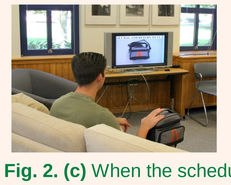


Fig. 2. (c) When the scheduled trip is approaching, Bob is reminded to carry the bag and needed items, e.g. bus pass, keys, iPAQ, meds, etc. **(d)** Bob checks-in the items for the trip (i.e., adds them to the bag) according to the inventory presented on the TV.

Fig. 2. (e) Following instructions on the screen, Bob has all items in his travel bag and is ready to start the trip.

Building the travel bag

Considering the accessibility, effectiveness, availability and cost, we selected a set of building blocks to create a bag system. The requirements for these blocks were obtained through focus groups, product surveys, pilot studies, and engineers with years of experience.

Table 1. Required hardware to implement the travel bag system

Interface	Functionality	Hardware	Customize
PC to TV	Remote control	IR module (USB)	VC
PC to TV	A/V from PC	Converter	COTS
TV to PC	Detect TV status	Standard	COTS .dll
Desktop Apps	Display received info on TV	Standard (RS232, Internet)	Java, C#, or VC
Bag/PC	Wireless link; also assure users are watching TV	Mica2	nesC
Bag/safety -watch	Wireless link to prevent bag from being lost	Mica2	nesC, HW design
Bag/RFID tag	Assure needed items are in the bag	RFID reader (RS232)	nesC
Power for bags	Ensure bag working reliably for days	Battery and converter	Customized converter

However, with the above as best choices, we find ourselves trapped in heterogeneity: different components use different languages and OS, and adopt different communication media and protocols.

The situation is more desperate from the viewpoint of personal and contextual requirements engineering, where requirements vary over time and context [1].

Survive with rich components

We advocate reuse through rich components to reduce development costs. Components are not accidentally but systematically built and reused.

Table 2. Some representatives from component repository for the bag system. These components satisfy most needs for communication between PC and motes listed in Table 1.

Components	Functionality
GPS-server	Broadcast GPS reading
Safety-watch	A watch-style wrist wrapper to safeguard the bag from being lost
Proximity component	Estimate distance between sender and receiver
RFID reader	Provide RFID tag reading
Mote-modem	Provide query-response messaging service
TosMsg serialization	Encode/decode TosMsg from serial communication
Timer component	Prevent time drifting and provide accurate timing after calibration
USART component	Provide printing, memory swapping service

Also, components should be built and reused at different scales. A component can be reused by a product or another component.

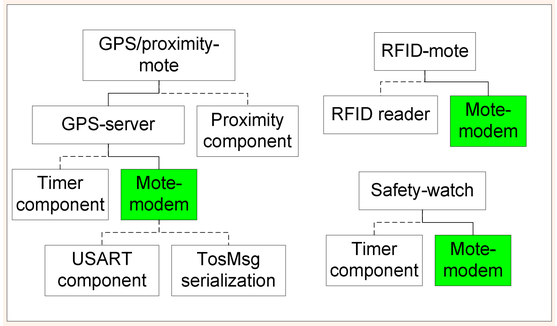
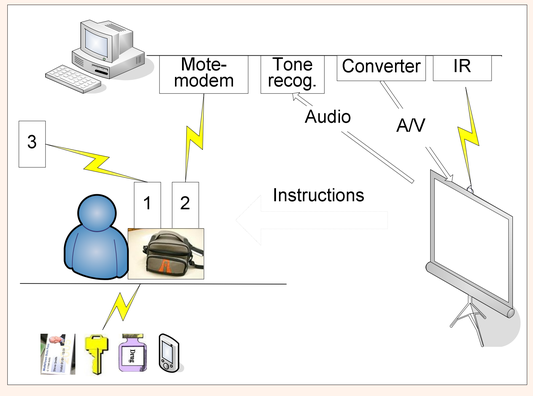


Fig. 3. Component ingredients for part of the bag. Dotted lines represent reuse of pure software; solid lines represent a product containing software and hardware. The reuse scale grows from bottom up.



Note:
1. GPS/proximity-mote; 2. RFID-mote; 3. Safety-watch
Fig. 4. With rich components, we have built a rapid prototype that enables the scenario in Fig 2.

In our experience, it took an *expert* Mote/nesC programmer about one week to build a communication path from PC-mote-PC that reported an RFID reading. Once we introduced rich-components, e.g., a Mote-modem, it took only about two days for a moderately savvy Java programmer.

When we take reliability into consideration, the saving is more significant. About four months ago our programmers had problems with a flakey GPS reading. It took more than a week to inspect the code and test the mote without any result. The problem became clear only after a hardware engineer found the cause to be a redundant power regulator on the circuit board. With rich components, e.g., a GPS-server, we can now avoid such traps by reusing well-tested components.

Conclusions

Wearable computing systems are composed of heterogeneous components. With the rich-components approach, we are able to hide heterogeneity behind simple interfaces, making components more pluggable and systems more quickly constructed.

We have found that in wearable computing systems requirements are dynamic [1]. Requirements may change because of changing user needs or because of changes in the environment. In some cases we can adapt to these changes by swapping components in a system. In other cases, we would like existing components to adapt to new requirements. The idea of self-adapting components that retain their reusability is a topic that we have just started to explore [2].

Finally, we are concerned about the safety and privacy of the wearable systems we build. In the past we have used formal models to prove that safety and privacy properties hold for the static systems we have built. However, the rich-component approach brings new challenges to the modeling effort; systems are expected to be dynamically changing. We are working on mechanisms that will allow us to quickly prove that our safety and privacy concerns continue to be met as a system evolves.

Literature cited

[1] Sutcliffe, A., Fickas, S., Sohlberg, M., Personal and Contextual Requirements Engineering, 13th IEEE International Conference on Requirements Engineering (best paper award), Paris, 2005.

[2] Fickas, S., Clinical Requirements Engineering, invited paper at 27th IEEE International Conference on Software Engineering, St. Louis, 2005.

[3] Chen, Z., Fickas, S., The Plain Old Television in a Smart Apartment, (invited paper), submitted to 1st IEEE International Conference on Collaborative Computing, Dec 2005.

Acknowledgments

This research was partially supported by an NSF ITR grant under number IIS-0313324 and Intel sponsorship. We thank our community partner, ShelterCare of Eugene.

For further information

Please contact *zbchen@cs.uoregon.edu or fickas@cs.uoregon.edu*. More information on this and related projects can be obtained at *http://www.go-outside.org/*