

Keeping Up With Demand: Supporting Community Appropriation Without Hacking

Derek Reilly

EDGE Lab

Faculty of Computer Science

Dalhousie University, Canada

1-902-229-1612

reilly@cs.dal.ca

1. INTRODUCTION

From the growth of text messaging to the adaptation of blogging in response to global events, technology appropriation can be very exciting, providing a medium through which new patterns of human behaviour emerge and old ones are adapted. Evolution of use can far outstrip the original intentions of a given technology. While it is important that a technology is useful in its original form, it can be more important that its larger potential is apparent to a community of users, and that there exists a way for those users to explore that potential.

It seems that many ubiquitous computing scenarios would benefit from systems designed to support both ‘hackability’ (and likely ongoing iterative design) and appropriation¹. A variety of approaches have been proposed to address the need for adaptability in ubiquitous computing. End user programming, often in a tangible format, has found a promising domain in ubiquitous computing, allowing users to express ad hoc needs [1]. Authoring environments such as the Mobile Bristol toolkit [2] allow non-programmers to quickly develop context-sensitive applications. Participatory design in ubiquitous computing [3] allows a target group to take ownership of the technology and so might promote its adoption and ongoing evolution.

The tension between concrete solutions and changing needs is a central theme in applications design. Christopher Alexander envisioned an organic approach to architecture, involving members of the community as “resident experts” to support the ongoing evolution of a physical space. His notion of design patterns has strongly influenced software development practice, and other aspects of his approach, including the direct involvement of members of the community, are reflected in joint application development (JAD) and work-focused design. However, the notion of “resident expert” as a member of the community tasked with evolving a system does not offer an

¹ I believe a distinction can be made between ‘appropriation’ and ‘hacking’. Appropriation involves using an artifact not expressly intended for the design space. Hacking takes an artifact originally developed for the design space, and modifies it to suit other purposes, usually related to the original intention. Appropriation may or may not involve hacking. I believe that history shows more success in community appropriation than in community hacking.

obvious parallel in traditional software process. More typically, maintenance is performed by programmers. This has direct implications for ‘hackability’.

The ongoing design and evolution of systems by communities themselves has long been a goal of many technologists and designers. Alan Kay’s Dynabook [4] articulated a vision of the computer user as active creator and sharer of computational objects. We see some of that reflected in end-user or domain-specific programming languages and environments from ToonTalk [5] to LabView [6]. However, except in a few cases such languages have failed to capture a strong user base. More modest approaches integrate programmatic capabilities into existing software systems like spreadsheets, or provide templates and wizards to automatically generate electronic artifacts complying with an underlying language (e.g. web authoring tools). ‘Programming by example’ attempts to eliminate the awareness of an underlying language by letting the user provide concrete examples as templates of desired behaviour. All these represent attempts to engineer extensibility or programmability into systems in such a way that non-programmers can more effectively adapt computer systems to their needs.

As study of these and other approaches show, it is most important to understand the community of users, their activity patterns and social structures, in order to support the need for computability/adaptability. The domain-specific programming environment LabView has been successful in employing a metaphor familiar to their target audience (circuit design). Macromedia has continually adapted their Flash [7] authoring environment to accommodate the desired capabilities illustrated through the ingenious hacks of their programmer community, that is by keeping up with their user base.

However, not all novel uses of a technology require this level of adaptation. Often the fundamental capability of a system provides a greater breadth of possible uses than the original intended use (such as the web). Communities do not in every case actively engage in changing the core system, but rather can find novel uses for the existing technology (appropriation), and express or make evident the need for modifications.

Even within the programming community, where ‘hackability’ finds more formal expression in ‘reusability’ or ‘extensibility’, the notion has detractors. The mixed success of object-oriented programming, component models, reusable libraries and

frameworks belies the fact that many programmers resent ‘fitting a square peg into a round hole’, just as many regularly opt for efficiency and singularity of purpose over reusability. Software design patterns attempt to address this by focusing on the abstract as reusable while the concrete (i.e. program) may not be. This approach may be valuable for ubiquitous computing systems.

I believe it would be extremely difficult to design a system that has immediate appeal to its community of users and would be able to accommodate all emergent needs and uses without some form of ongoing input from designers, and modification of aspects of the design as a whole. As the following example illustrates, our design approach emphasizes appropriation over hackability, by developing a system suited to a specific need, which is abstract enough to exhibit a potential for use in other domains by a larger community.

2. DATA COLLECTION IN PARKS

To avoid meticulous and slow-moving data collection, parks today often sacrifice awareness of subtle changes, and of variations across a finer granularity of space. Parks use remote sensing, sampling and aggregation to build high-level views of spatio-temporal phenomenon such as forest growth, the activities of backcountry visitors, and wildlife populations. When more precision is desired, techniques such as adaptive sampling can be applied. Such methods maintain a "top-down" focus, increasing the resolution of data capture only where deemed necessary. Such an approach is suitable in many circumstances, but cannot answer every information need.

More detailed observation often involves lengthy and tedious data collection in the field, combining GPS and sensor data with standard desktop computing software, or specialized software derived ultimately from the desktop computing paradigm. Data collection of this sort can require a significant portion of the time of an individual hired for skills in other areas (such as conservation, geological research, etc.).

Our current research endeavors to permit a "bottom-up" approach to data collection by simplifying the data capture task at the level of the individual. We are exploring the use of glyphs (symbolic language) to record location-specific observations. Reducing the data entry burden allows the possibility of a richer array of information to be maintained, a more dynamic information base, and finer granularity for manually entered data. Such an approach may complement the existing techniques for data collection, and open up Geographic Information Systems (GIS) use to more prosaic or everyday endeavors such as recording litter, and bird sightings by amateur birders.

We believe that symbolic communication holds promise for location-specific information capture, and may encourage appropriation. Beginning with scientific data collection in parks, we can imagine evolution of the design and movement into other areas, as evinced in following scenario:

A rock climber is scaling a cliff for the first time. When she encounters an area of loose rock, she makes a gesture encircling the area and draws an X through it with her hand, indicating both the region and the category of information. This warning becomes available to other climbers.

At this stage of the research, we are particularly interested in how a language of symbols might be designed and evolved by a community of users. Adoption of symbolic language is far from guaranteed. For example, warchalking (a simple glyph-based language for advertising WiFi connectivity) may have captured the imagination of many but was apparently used by very few. Development of a system for scientific data collection in parks will involve participatory design to establish a basis, but must then be observed in context to understand adoption and possible appropriation to other uses.

3. SUMMARY

To achieve ongoing community appropriation, we can first create a useful artifact/system which in its design the potential for modification and embodies a promise that is tied to the community's own creativity. Such an artifact would preferably be designed with members of the community to foster ownership and focus the design. At least some of those community members involved in design should have a penchant for tinkering and innovation. Finally, it is likely necessary to have ongoing engagement by technologists as appropriation evolves, who can then provide more direct support for needs as they emerge.

4. REFERENCES

- [1] R. Hague, A. Blackwell, and P. Robinson, End User Programming in the Networked Home, *Proceedings of the 1st Equator Workshop on Ubiquitous Computing in Domestic Environments*, 2001, pp. 46-67.
- [2] R. Hull, B. Clayton, T. Melamad. Rapid Authoring of Mediascapes. in *The Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*. 2004.
- [3] J. Montemayor, A. Druin et al., Physical Programming : Designing Tools for Children to Create Interactive Environments, in *Proceedings of Human Factors and Computing Systems (CHI 2002)*, 2002.
- [4] A. Goldberg, A. and A. Kay, Personal Dynamic Media , *Xerox PARC Learning Research Group Technical Report*, March 1976.
- [5] K. Kahn, “ToonTalk - An Animated Programming Environment for Children”, *Journal of Visual Languages and Computing*, 1996, 7(2): pp.197-218.
- [6] <http://www.labview.com>
- [7] <http://www.macromedia.com/flash>