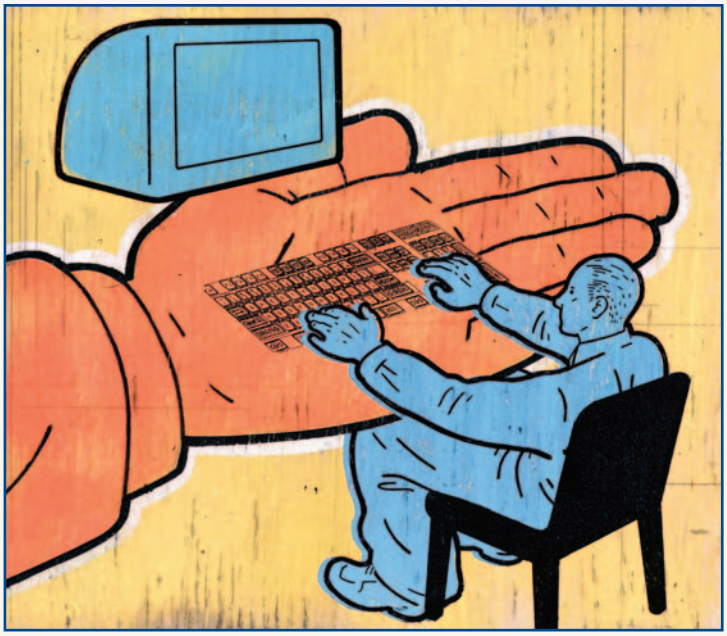

How to Provide Useful ICT When Called Upon

Edwin Blake
University of Cape Town
edwin@cs.uct.ac.za

If a government or NGO (a nongovernmental, nonprofit organization) of a developing country decides that information and communications technology (ICT) can be useful for development, the question that arises for practitioners is how to provide a useful sustainable system that is wanted and actually used. Since computers have been regarded as devices that disempower marginalized people, we want to empower disadvantaged people to use and control the technology and avoid creating passive consumers. As academics we want to train ourselves, our students, and our industry to provide ICT in that fashion.

I am arguing for a method for investigating these situations, not a method for solving them. I do not think we are yet ready to uncover the differences faced with developing ICT for developing countries in a systematic way. I doubt that such an exercise is really possible: It would imply that one categorizes issues in the developing world as characteristically different than in other situations. For example, I would certainly not try to suggest the kind of technology that developing-world applications should use. It could be old, established technology, but frequently the most advanced technology is the easiest to use. Instead I am striving to develop a method whereby such differences can be uncovered for a particular chosen problem domain by developers who are trying to make a difference.

In order to investigate this issue, my colleagues and I adopted a method whereby we build systems and then reflect on the design method that enables us to have a useful and sustainable impact. We are coming up with a *Socially Aware Software Engineering* approach that is based on critical action research: fostering change in a

Editor: Gary Marsden • University of Cape Town • gaz@acm.org

community through facilitating action. A key realization has been that the cycles of action research map well onto the cycles of iterative software development. The software serves to document and embody the learning that occurs as the action research investigation proceeds. Our approach employs user-centered methods from HCI, including participatory design to ensure solutions meet user requirements, while the action research cycles guide the process of working with the communities.

Our work has included creating a field computer that allows semiliterate animal trackers to record their field observations, a computer-assisted system to bridge communication between members of the deaf community and the hearing, and a health consultation system for remote rural communities. The health consultation system made use of voice-over-IP technology to enable nurses at a rural clinic to consult with doctors at the local hospital in order to provide improved health care to local people. An intriguing feature of rural areas in South Africa is that people are not concentrated in small towns and villages but rather are spread out across scattered settlements throughout the countryside. Rural hospitals tend to support ten to 12 satellite clinics in a roughly 20-kilometer radius. Each clinic then supports a somewhat dispersed population of up to 20,000. We built a long-range WiFi network and designed a mixed synchronous (real-time) and asynchronous (store-and-forward) communication system to support remote tele-consultation. The asynchronous feature requirements emerged from the end users as the researchers learned that frequent power outages, and, more importantly, overburdened doctors and nurses, meant that end users could rarely talk in real time when they wanted to.

Now in the third year of active field trials, consisting of a series of Action Research cycles, we have come to focus on the human-computer interface. The long-term iterative process allowed the target users to participate and guide the development of the system, even though they largely remain limited in ICT skills. We have incorporated a continuing ICT training element to the process in order to move toward the goal of employing more and more Participatory Design techniques as user skills improve.

We have tried to use Participatory Design to allow the end user to participate in the software-design process. This method as it stands is situated in developed economies where users have some degree of reflective sophistication about their work in relation to the possibilities offered by ICTs. This has two flaws from my point of view:

Flaw 1: Participatory Design assumes the user community knows about technological possibilities and limitations at the most basic level. Our communities have a patchy knowledge; they may have neither electricity in their homes nor a fixed-line phone, but they may well have used a mobile phone.

Flaw 2: Participatory Design assumes that software designers can bridge cultural and linguistic gaps between themselves and their target communities. However, these gaps can be enormous. The technological requirements exist within a complex web of other needs, relationships, and societal obligations. Misinterpretation (on both sides) and unexpected needs are common. It is difficult for IT practitioners to appreciate, for example, how an IT empowerment exercise may threaten power relations in such communities with dangerous consequences for several participants.

Our tentative solution to this is twofold:

1. Find local "interpreters" or champions who can bridge the gaps. Students from these communities can also assist. Such people act as our intermediaries into the communities.

2. Educate interested members of the target communities in the use of ICT and expose them to technology even if the use they make of the technology is not (initially) related to the problem area we are trying to address.

Action Research cycles mutually educate both the community and the ICT developers on the problem domains and the relevance of technology to that domain.

I now believe that software engineering as a profession has to change to translate the social and economic needs of local communities into useful systems. A *Socially Aware Software Engineering* method that extends beyond the purely technical is necessary. Ethics focused on dealing with development priorities has to replace the emphasis on First World professional issues and values. Professionals should accept a new approach that involves the codevelopment of applications with communities in a socially sensitive fashion. Universities (and NGOs) have great opportunities to design and implement new approaches to using technology to support local communities in developing countries. ✦

ABOUT THE AUTHOR *Edwin Blake is a professor in the department of computer science at the University of Cape Town, South Africa. He received BS degrees from the University of Cape Town and the University of Witwatersrand, South Africa, and a PhD from Queen Mary College, London University, UK. He specializes in human-computer interaction, interactive computer graphics, virtual reality, as well as the use of IT for socio-economic development.*