PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

FirstCite® e-publishing

Phil. Trans. R. Soc. A
doi:10.1098/rsta.2008.0119
Published online

People are people, but technology is not technology

By Gary Marsden*, Andrew Maunder and Munier Parker

Department of Computer Science, University of Cape Town, Rondebosch 7701, Republic of South Africa

Ubiquitous computing is about more than having multiple computers in our environment; it is also about computers venturing into completely new environments. In this paper, we examine the impact of computers in the developing world and look at why most interventions to date have failed to address the key needs of the users and their context. Through an analysis of existing software design techniques, and various case studies, we propose a new model for software creation, which we believe will address the issue of creating technologies for developing world nations.

Keywords: developing world; cellular telephones; human–computer interaction

1. Introduction

The common theme in this series of articles is about the increasing ubiquity of computing devices. As Moore's (1965) Law continues unabated, we tend to imagine a future wherein more and more of the artefacts that surround us become 'smart'.

Besides deepening and entrenching computing devices in our environment, a less-studied consequence of Moore's Law is the increased reach of ICT into new environments, specifically the developing world. At present, it is economically viable to manufacture and distribute a cellular handset for $30.[1] Perhaps surprisingly, the fastest growing cellular market in the world right now is Africa (ITU 2008, http://www.itu.int/ITU-D/ict/statistics/ict/index.html); many people (or communities) are able to afford a handset that can be used to overcome all sorts of socio-economic issues. Of course, they can also use it to chat with their friends.

We, as technology designers, tend to forget that last point, yet it is what inspired the title for this talk. People living in developing regions, such as Africa, are still people and want to talk to their friends and families, much as people from regions with sophisticated ICT infrastructures. However, the understanding of ICT and its needs for people in the developing world are hugely different from those people who live in a technology-rich environment. The rest of this paper

* Author for correspondence (gaz@cs.uct.ac.za).

[1] Based on current (January 2008) figures from South Africa.

One contribution of 19 to a Discussion Meeting Issue 'From computers to ubiquitous computing, by 2020'.

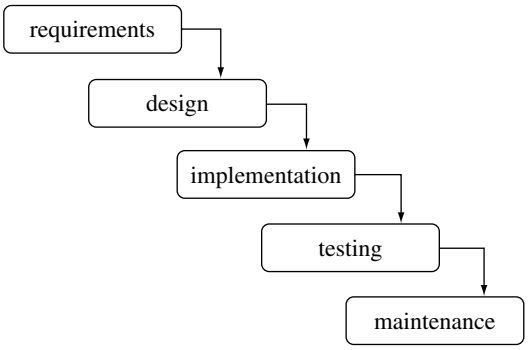

Figure 1. A standard waterfall model for software creation.

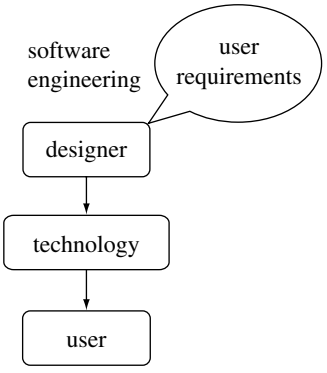

Figure 2. A software engineering view of human–computer interaction.

explores how to create technologies that are appropriate to people living in ICT-sparse contexts. In addressing this topic, we will look at various models for creating appropriate technologies and discuss some failed and successful applications of technology.

2. The science of users

In the 1960s and 1970s, interactive computer technology started moving from research laboratories into the wider commercial world. If we look at the types of uses these computers were put to, we see that initial adoption was in the insurance and banking sectors where employees were completing well-defined tasks, e.g. calculating insurance risk (Landauer 1997).

Methods for developing software created in this era have a linear structure, as exemplified by the 'waterfall' model (McConnell 2004). As can be seen from figure 1, the process starts with the 'requirements'; the model is predicated on the fact that the requirements are given from which the rest of the process can proceed to optimize for those requirements. In figure 2, we simplify this diagram to show how the system designers relate to the users. Essentially, this only happens in the form of the user's task requirements that come from some third party; the only link between technology creator and technology consumer is the technology itself.

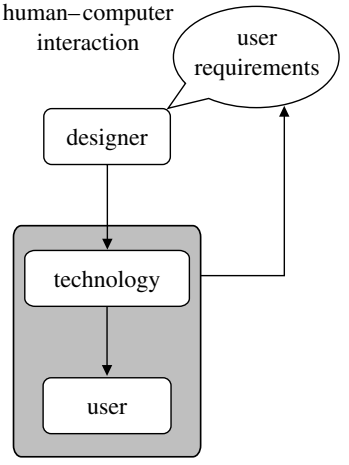

Figure 3. A standard HCI model.

To further improve the integration of computers into a commercial environment, various studies were conducted on user efficiency and productivity while using these office automation systems. As more studies were undertaken, the academic discipline of human–computer interaction (HCI) slowly developed to measure the efficiency and effectiveness of the interaction between the computer and the user. Typically, these studies took the form of a psychological experiment in which the interaction between the users and the computer was observed in laboratory conditions. The results from these studies were analysed to better understand the human cognitive processes and better optimize the interaction between the user and the computer. A typical paper in this genre would be Card *et al.*'s (1980) work on the time taken for the users to complete a task when selecting from options on a screen. We may summarize this process as shown in figure 3, wherein the designer has a better understanding of how to design technology based on observing the users interacting directly with a system.

Currently, as this issue has highlighted, computers are available to more than just office workers and are becoming involved in every aspect of human life. The former HCI measurements of effectiveness and efficiency are not always relevant now. In the era of ubiquitous computing, researchers are interested in understanding the more subtle relationships between humans and digital technology.

In order to create more relevant technology, HCI has adopted methods such as participatory design (Schuler & Namioka 1993), wherein the end-users of the technology become co-designers and work alongside the design team from the initial concept sketch to the final system evaluation. Participatory design is just one technique in a wider scheme of the user-centred design, wherein the user is placed at the centre of the design process and prototypes of a system are rapidly created and evaluated with the end-user. This is possible as most users now have at least a vague understanding of digital technology and the sorts of things that computers can do well. Designers and users can share ideas and create prototypes jointly. These prototypes can be everything from a paper sketch through to a computer program; the point being that the designer and the user have a shared artefact through which they can communicate. This process is outlined in figure 4.

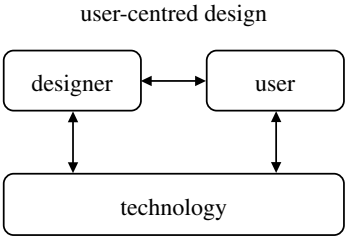

Figure 4. A standard user-centred design model.

There has been much written about the success of these processes (Muller 1991), but all of them are predicated on the fact that the users involved have a good understanding of what digital technology can achieve. In short, these techniques facilitate the deepening and entrenchment of technology with digitally literate users. The challenge now facing HCI, however, is how do we design appropriate digital technology for those who do not know what digital technology is?

3. Back to basics

Within HCI, there has been much work on the 'internationalization' of software. For example, a key book by del Galdo & Nielsen (1996) looks at the types of problem that software vendors face when selling technology into new markets, where language and visual literacy is very different from the standard US English.[2] The book talks about the need to translate the interface and tackle issues such as redesigning icons. However, this work is built on the 'office-worker' model of the user, where it is assumed that the technology will be used in some structured environment to achieve some task. The result is that most of this work treats the technology as an artefact that needs to be optimized for local conditions.

At the other end of the scale are social scientists, such as Hofstede, who tackle the internationalization problem by analysing the users to tease out different behaviour patterns from different cultures. The most popular exemplar of this work is Hofstede's book (1997), which distils global culture into five axes. The book comes with tables containing plot points for most countries so that outsiders can gain an understanding of their target culture. As an example, the UK scores low (only 35 compared with Greece's top score of 114) on the uncertainty avoidance scale; apparently British people can live with a lot of ambiguity in their lives. But people are individual people and not an 'average'; for example, some days we feel more risk averse than others. And, does everyone living in the UK really feel the same way about risk? Worse still, there is much evidence that the use of communication devices engenders a particular culture among their users, separate from the 'home' culture that the user may originate from (Berg *et al.* 2003). So while cultural models are interesting and may give a broad-brush insight into certain societies (Marcus & Gould (2000) used them to

[2] 'Standard' here refers to the de facto standard that most software is created by companies that communicate in US English.

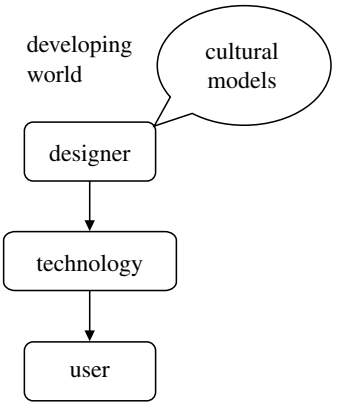

Figure 5. A design process based on cultural models.

analyse websites from different cultures and compared them on cultural axes), it is not clear how this knowledge might be directly leveraged to create better technology. And if the research on device culture is to be believed, we do not even know what that culture will be until the device is introduced to, and 'domesticated' by, the users (Silverstone & Hirsch 1992).

Ultimately, however, whether one is following internationalization guidelines or cultural models, the design process is still disassociated from the end-users, as shown in figure 5. In fact, comparing figure 5 with figure 2 shows that the state of the art in creating software for the developing world is to follow models of HCI from the 1980s!

There are many reasons for this, but the primary one is that user-centred techniques that the rest of the world is using simply do not work in the developing world. There have been attempts to make them work, but all report failures at some level, e.g. Medhi *et al.*'s (2006) work on interfaces for illiterate users.

(*a*) *User-centred design in the developing world*

As was stated before, a user-centred design will work only when the users have an understanding of what is possible with digital technology. For example, one project we worked on in the Eastern Cape (a very poor and rural part of South Africa) was to create software for nurses to use in a clinic. We started out following standard UCD procedures by encouraging one of the clinic workers to take part in a participatory design session. This involved asking her what she wanted from a computer system and asking her to make paper sketches to reflect these ideas. However, she was confused about the purpose of the paper; she had seen computers before but did not understand that software was a mutable entity that could be altered in the same way as the paper sketch. To these users, adding 20 more USB ports to the computer was as easy as adding new menu options to the software; the computer (hardware and software) came out of the box as-is and there was nothing that could be changed about it. The perception was that we were wasting her time.

In the end, reasoning that something was better than nothing, we set about implementing an initial prototype system. We hoped that, by observing the system in use, we could elicit comments from the nurses and improve the design,

using the artefact to seed the culture of its use. In order to improve the quality of the initial system, our research group partnered with a charitable[3] organization called bridges.org that was investigating how ICT could best be used for upliftment in developing countries. This organization had studied several ICT interventions in an effort to draw out a set of success criteria for future projects to follow and consider (Maunder *et al.* 2006).

After building the initial system, it took a further six months' training of the nurses before they were in a position to use the software as part of their daily routine (they had to be taught the basics of mouse movement, files, windows, etc.). After a few trials, it became clear that there were usability problems with the system. When we told the nurses that we were going to fix these, they became highly concerned. They had undergone six months of training to use the system and now we wanted to change it! We were wasting their time again.

We tried various approaches to improving the system. One was to iterate the design geographically rather than temporally—we would deploy the next version of the system in the neighbouring clinic and so on (there were 10 similar clinics) until we arrived at a solution. This obviously introduces problems between the clinics and creating new software for groups of two users is hardly a sustainable development model given the ICT requirements in the developing world.

4. Towards UCD4D

From the systems we have built (Jones & Marsden 2004; Maunder *et al.* 2007), it is clear that digital technology can positively affect the lives of those living in developing countries. At present, an estimated 77 per cent of South Africans have a cellular handset (this in a country where only 11% earn enough money to be registered for income tax; CIA 2008, https://www.cia.gov/library/publications/the-world-factbook/index.html). The motivation for using these devices is clearly evident and their ubiquity means that handset-based software can be deployed to a large percentage of the population. So, if the hardware platform is in place, do we now have a model for creating the applications that will improve the lives of their users?

(*a*) *Human access points*

The problem we had with the nurses still remains—it is hard to conceptualize what technology might be able to do for you if you are not familiar with what it does or how it is created. However, there is something that happened when working with the nurses that gave us an insight into a way to take a user-centred approach to creating software for technically illiterate users. We eventually came across the husband of a doctor, originally from The Netherlands, who had been working in the community for many years. He was completely familiar with digital technology but was also sensitive to the needs and context of the people working in the clinic. He had also been training a local woman to use the basic functionality of a PC. Over time, we slowly learnt to use both of them as a proxy or, more crudely, a human access point (HAP) into the wider community. We could use them for design ideas and initial testing of the prototypes.

[3] Also known as non-governmental organizations or non-profit organizations.

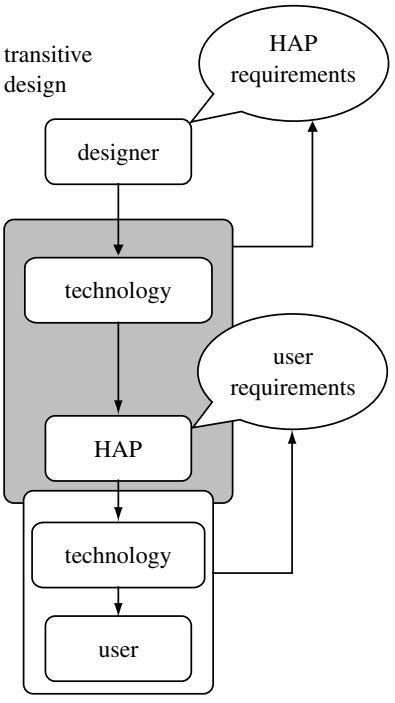

Figure 6. Using the 'HAP' as a user proxy.

This allowed us to create a more appropriate prototype than we could have created on our own, and from there, go on to trials with the target users. When those users could see what the technology did, it was possible for them to become more active participants in the design process. And finding this HAP was not an isolated incident.

On many of our projects, we have found at least one person in the community (perhaps a student who was given a bursary and sent to a school that had computers) who could act as this access point. Reflecting on our experiences, we realized that these people were more than just a way into the community for our research, they were the people who should be creating the technology for the users in the first place. Furthermore, this transitive investigation overcomes ethical issues incurred by researchers from outside a community observing and documenting processes and rituals that the community would rather keep as private.

Taking a look at figure 6, one will see that our current model of working is to put the HAP at the centre of the process, creating technology that allows the HAP to create solutions for his or her community. But what does that actually mean in practice?

(*b*) *Communitization*

Current software entertains a notion of 'personalization' wherein the user can, say, change the colour of text highlighting or the order email headers appear in a list. At the other end of the customization scale is open-source software that can

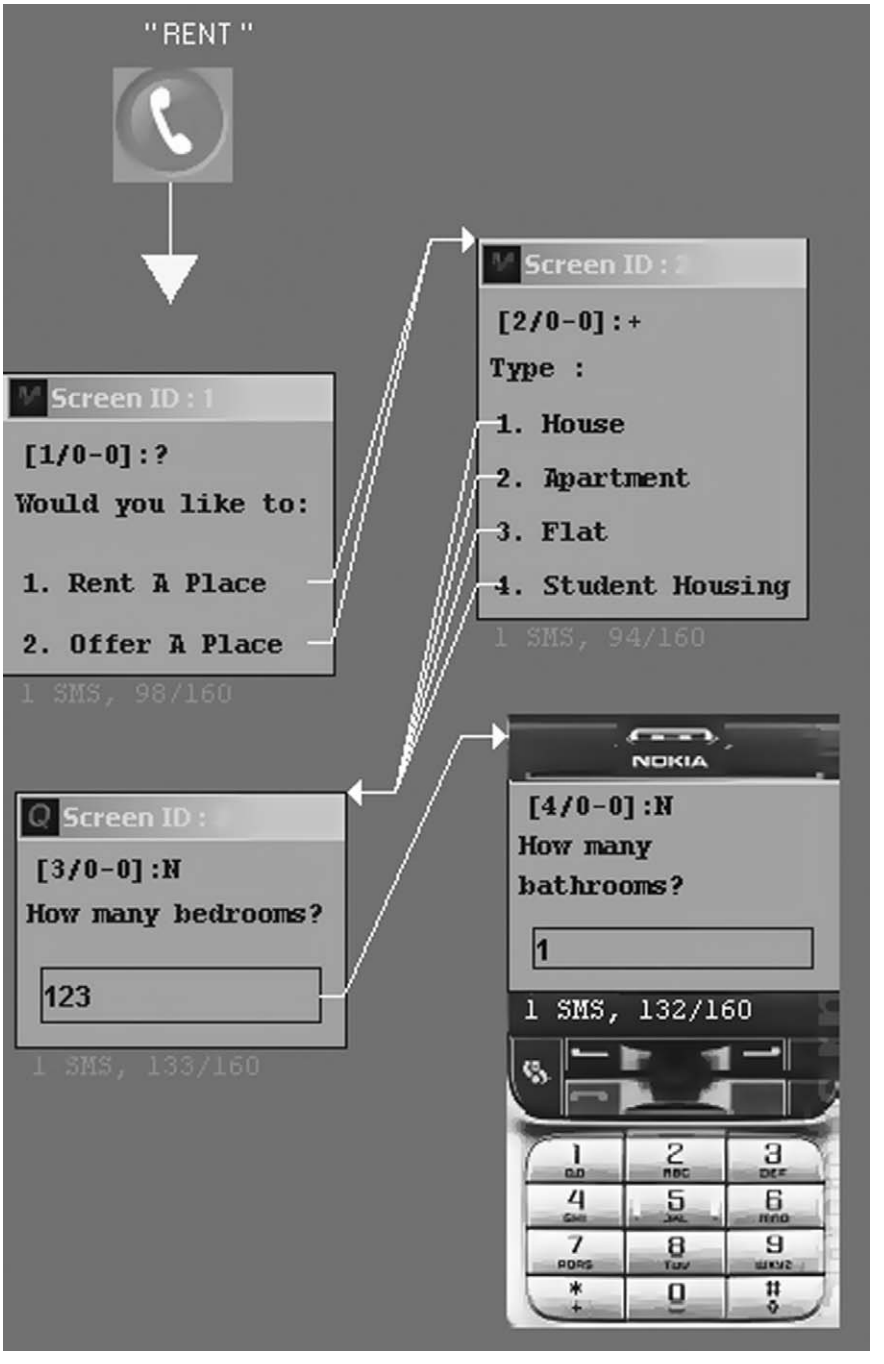

Figure 7. Software for creating information-gathering applications.

be configured and rewritten endlessly by anyone with several years' experience of programming in a high-level computer language. Alas, while our HAPs may understand the application of technology, few have the skills to adapt open-source applications to their needs.

We therefore propose the creation of 'communitizable' software, such as personalization, but for a community's use rather than an individual's use. To give an explanation of what we mean, consider the following project.

(*c*) *Data gathering*

A large part of caring for patients on anti-retroviral medicine (for the treatment of AIDS) involves monitoring their symptoms on a regular basis. In South Africa, this work is often undertaken by health workers; people from within a community who had been given some basic medical training. The data collected by the workers are fed back to the doctor and, based on those data, the doctor prescribes appropriate medicine.

This data collection process could be depressingly slow; many of the health workers have low literacy rates and the forms were complex. However, a Cape Town-based charity, Cell-life, created a piece of software that allows health workers to capture patient data on a cellular handset and SMS those data directly to the doctor. This system was so effective and so popular that many other groups (health care, government, charities, etc.) contacted Cell-life and asked for a modified version of the software.

One of the people involved in creating the original software realized that a single charity could never meet these various demands, so set about creating a piece of software to create this type of data collection application. The goal was to allow a domain expert (e.g. a doctor) with basic IT skills to create and deploy the application with no external help. After interviewing 150 users from all over Africa, a piece of software was created which allows the users to draw handset screens depicting a flow of interaction and data collection (figure 7). After the user creates the screen layouts, he or she clicks a button labelled 'Deploy' and the application is compiled and ready to run.

This software now makes it possible for any organization to use standard cellular handsets to act as data collection agents. There is no need for external software engineers or even HCI specialists to be involved. The agency that created the application can monitor it in use and modify it to better suit its environment.

5. Conclusions

Ubiquity is about more than the technological saturation of environments in highly developed countries. In the form of the mobile handset, digital technology is finding a foothold in parts of the world which may not even have reliable electricity supplies.

Many have tried to use existing HCI methods to create technology for the developing world. This approach comes unstuck as the developers and the users have fundamentally different understandings of the value of different aspects of digital technology; technology is not technology. Yet, this does not imply that the users in the developing world are different people from those in the developed world; people are still people who wish to use digital technology to enhance the way they live their lives, even if those lives are lived in very different contexts. Understanding the differences in context has been the problem, a problem that HCI techniques do not fully address.

One consequence of applying these HCI processes in this new context is that the users will need to be given sufficient literacy training so that they can take part in the process. Yet the goal of an HCI process should be about tailoring the ICT to the user; any unnecessary intervention with the user is clearly contrary to the original aims of the process.

Rather than affecting the end-users, we realize that, within most communities, there are people with a vision for how technology can best be used within their context. Our current efforts are therefore concentrated on creating communitizable software; empowering the community to create and refine its own digital technology.

The authors would like to thank Telkom/Siemens Centre of Excellence, Cell-life, the South African National Research Foundation and Microsoft Research Cambridge, which have all contributed to the funding of this research. We would also like to thank Edwin Blake and Bill Tucker for supporting and mentoring our work, as well as Susan Dray and Nic Bidwell for their helpful comments on early drafts of this work.

References

Berg, S., Taylor, A. S. & Harper, R. 2003 Mobile phones for the next generation: device designs for teenagers. In *Proc. CHI, Fort Lauderdale, Florida, USA, 5–10 October 2003*, pp. 433–440. New York, NY: ACM.

Card, S., Moran, T. & Newell, A. 1980 The keystroke-level model for user performance time with interactive systems. *Commun. ACM* **23**, 396–410. (doi:10.1145/358886.358895)

del Galdo, E. & Nielsen, J. 1996 *International user interfaces*. London, UK: Wiley.

Hofstede, G. 1997 *Cultures and organizations: software of the mind*. New York, NY: McGraw-Hill.

Jones, M. & Marsden, G. 2004 Please turn ON your mobile phone—first impressions of text-messaging in lectures. In *Proc. Mobile HCI, Glasgow, UK, September 2004*, pp. 436–440. Piscataway, NJ: IEEE.

Landauer, T. 1997 *The trouble with computers*. Cambridge, MA: MIT Press.

Marcus, A. & Gould, E. 2000 Crosscurrents: cultural dimensions and global web user-interface design. *Interactions* **7**, 32–46. (doi:10.1145/345190.345238)

Maunder, A., Tucker, W. & Marsden, G. 2006 Evaluating the relevance of the 'real access criteria' as a framework for rural HCI research. In *Proc. CHI-SA, Cape Town, 25–27 January 2006*, pp. 75–79.

Maunder, A., Marsden, G. & Harper, R. 2007 Bluetooth interaction with situated displays. In *Proc. Mobile HCI, Singapore, 9–12 September 2007*, pp. 188–191. New York, NY: ACM.

McConnell, S. 2004 *Coce complete*, 2nd edn. Redmond, WA: Microsoft Press.

Medhi, I., Sagar, A. & Toyama, K. 2006 Text-free user interfaces for illiterate and semi-literate users. In *Int. Conf. on Information and Communication Technologies and Development, Berkeley, USA, May 2006*, pp. 72–82. Piscataway, NJ: IEEE.

Moore, G. 1965 Cramming more components onto integrated circuits. *Electronics* **38**, 114–117.

Muller, M. 1991 PICTIVE—an exploration in participatory design. In *Proc. CHI, New Orleans, Louisiana, 27 April–2 May, 1991*, pp. 235–241. New York, NY: ACM.

Schuler, D. & Namioka, A. (eds) 1993 *Participatory design: principles and practice*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Silverstone, R. & Hirsch, E. (eds) 1992 *Consuming technologies: media and information in domestic spaces*. London, UK; New York, NY: Routledge Press.