Content Development Through the Keyhole

Alan Dix
Computing Dept., InfoLab21, Lancaster University, UK
http://www.hcibook.com/alan/
alan@hcibook.com

Ramesh Kozhissery
HP Labs, Bangalore, India
ramesh.kozhissery@hp.com

Ramprakash Ravichandran
HP Labs, Bangalore, India
ramprakash.r@gmail.com

Dinoop Dayanand
National Institute of Design, Bangalore, India
dinoop.d@gmail.com

http://www.hcibook.com/alan/papers/EISE2009-Keyhole/

A large proportion, if not the majority, of the world's population, including those in large parts of India, China and Africa, will experience the Internet and computing solely through a mobile phone. Furthermore in rural settings there may be no possibility of access to a PC or laptop – everything must happen through the mobile and many of the end-users may be illiterate. However, the applications available are typically simply those developed for western urban users. If these 'next billion' users or those close to them had the right tools they could create their own content and applications, better suited to their needs. However, current environments for production of web sites, programming and even plain content management, all assume that the developer or content author has a large display, not the tiny screen available on a mobile device. This paper discusses whether it is possible to create rich content or to code using such a device, to create content or applications 'through the keyhole' of the mobile phone screen.

Keywords*: near-end-user programming, content production, mobile-phone applications*

1. BACKGROUND AND MOTIVATION

A large proportion, if not the majority, of the world's population, including those in large parts of India, China and Africa, will experience the Internet and computing solely through a mobile phone. Furthermore in rural settings there may be no possibility of access to a PC or laptop – everything must happen through the mobile and many of the end-users may be illiterate. The applications commonly available on mobile platforms have been driven by expectations and requirements of Western countries and/or more affluent parts of the urban population, completely different cultures and social environments to those of this new 'next billion' users. More appropriate applications can be created centrally as has been the case with traditional applications, but it is likely that many of the needs will be locally or linguistically specific. It is those closest to the point of use who are likely to understand the needs and requirements best, but are themselves unlikely to have access to the high-end computers with large displays for which content development and programming environments are currently designed. Is it possible to develop content close to the point of use using available mobile technology?

In this paper we will explore this issue, first of all framing the broad problem area (section 2 and 3), and then exploring a few points in the resulting design space in greater detail. In particular, we will present a detailed scenario for tiny-screen community content production (section 4) and also look at what a tiny-screen programming environment might be like (section 5). Our aim is not to produce the best design in either case, but rather to demonstrate that it is possible to envisage means for radical end-user or near-use development even when the device available does not have a PC-sized screen.

Where appropriate we will draw particularly on the Indian context of use, but most of the issues that India faces (multiple minority languages, widespread illiteracy and poverty) are similar in other emerging markets.

2. BACKGROUND AND MOTIVATION

This issue was first raised at a group discussion at the Mobile Design Dialog (MDD) conference in Cambridge in April 2008 [15] and was subsequently the brief for 'bootcamp' sessions at the Winter School on Interactive Technologies in Bangalore in February 2009 [23]. At MDD one of the participants described a small businessman in rural Africa who used the address book on the phone to keep accounts of who owed money for goods – the 'phone number' field for each person was instead the amount owed. It is hard for those both geographically far away, or even in the same country but in a different social situation, to anticipate what applications will be of value in particular situations and so ideally those closer to the actual situation should be able to create applications or content suited to local needs.

In a report on one of ActionAid's projects, Beardon et al. [2] say:

> "*ICTs can be used to strengthen local traditions and cultures of communication, but only by design: people need to appropriate the technology, and give it functions which suit their needs and motivations.*"

This highlights the need for highly participative design processes when the expertise comes from outside, but also suggests the potential for 'home grown' ICT solutions. Experience in traditional computer systems certainly demonstrates the value of such systems: VisiCalc, the first spreadsheet, was designed by and accountant and a 'techie' together, and most of the popular software for doctors in general practice in the UK was developed out of bespoke programs on home computers built by GPs initially for their own use. In recent times web software (such as PHP) and web2.0 services (such as Facebook) have often emerged from a "build-it for myself and friends" philosophy. Experience of projects such as the "Hole in the Wall" project [14], which inspired the writer of Q&A (aka Slumdog Millionaire) suggests that given suitable resources those with even the most deprived backgrounds can rapidly learn to appropriate computers. In an interesting turn-about Professor Mitra is now adopting the same ethos he developed in Hyderabad to deprived areas of Newcastle [20].

The value of near-end-user development was demonstrated clearly by Marsden et al. at the University of Cape Town [13]. They had noted the need for nurses to create phone-based surveys (in the initial example for AIDS), but the nurses themselves did not have the skills or knowledge to use content development tools and had trouble conceptualising possible applications even in participatory design sessions. Happily, they eventually found someone in the local community, the wife of a doctor, who had sufficient knowledge of IT to be able to appreciate its potential, and sufficient understanding of the local needs to be able to address them. Marsden et al. describe these people close to the point of need as "human access points" (HAP), often someone with a higher level of education than typical of the area (maybe high school), but with understanding of local conditions and needs. It was for this HAP that they targeted the design, a PC-based graphical authoring system to produce simple surveys, and this was subsequently used to design the survey used for a substantial data gathering exercise.

In the above example, the local expert (HAP) was assumed to have access to a PC for authoring surveys even though the delivery platform would be a mobile phone. Similarly development environments for generic programming (e.g. Eclipse), scripting (e.g. Flash), and content (e.g. Dreamweaver) all assume a PC with a large screen even if cross-developing for a small-screen end-user device. But, as we have noted, in rural settings even this may be out of reach – the only means of access is the mobile phone. Is it possible to develop effective means for local experts to develop content or even code through the 'keyhole' of a mobile phone display?

3. CONTEXT: PEOPLE AND TECHNOLOGY

Roles

We can identify three main roles in the development of more appropriate systems for these emerging user bases:

- *infrastructure designer and developer* –Those who develop the underlying infrastructure and application shell, maybe central resources (e.g. web servers), and may have influence over service provision. In the example above this would be the University of Cape Town team. For this role, we assume few if any limits to the hardware and software available.
- *local developer* – The local expert or HAP who will use the provided infrastructure to provide content, applications or services. May be assumed to have higher level of education and access to higher levels of hardware (e.g. web-enabled phone).

- *end-user* – The person who is the final user of the system. May be illiterate or speak minority language and may have only basic knowledge of operating the device. (Although see later, there may be a further group of content consumers.)

In this paper we are effectively taking the role of the infrastructure designer and developer, and so we focus solely on the local developer and end user only. We therefore will drop 'local' and simply refer to the local expert as 'the developer'. However 'developer' will be taken to broadly include content production of all kinds, not just coding. For example, the survey production system was somewhere between pure content production and basic scripting as conditions could be used to create multi-path surveys.

As well as these three key roles, there may also be non-local end users who have access to more sophisticated technology. This would occur, for example, if the application involved some sort of advertising/sale for local produce or was simply offering news, experiences of the local community on the web.

These non-local users highlight three different types of usage scenario:

- *local individual use* – use by one person with a phone, for example, obtaining agricultural information or managing small accounts
- *local collaborative use* – local sharing and reminiscing, for example capturing local folk stories, craft knowledge
- *non-local use* – publishing information potentially to the world, for example village news or internet selling

Technology patterns

We can consider a number of levels of technology for mobile devices:

1. basic voice phone + IVR (use of digits to select services)
2. SMS/MMS with camera and media
3. WAP/web-enabled phone
4. smart phone (downloadable applets etc.)

With these in mind we can consider different situations in terms of the assumed level of technology available to (a) the local expert and (b) the local end users. Clearly the answer to this will vary over time as both the penetration of phones and the sophistication of base-level phones increase. At present (3) is about the best we can reasonably assume even for local experts, but it is reasonable to assume that (4) will become commonly available for local expert use within the next 5 years. The kinds of technology uses are also likely to vary greatly between areas and to some extent between individuals; however, the fact that phones are often shared or used by other than their owners, means we do not have the simple equation of phone owner = phone user assumed in most western phone applications. We will however assume that the local expert has at least the same level of technology and probably higher than the local end user.

Note that in all cases except (4), rich-media production is likely to be a problem even when the phone is equipped with a digital camera or is capable of audio or video recording. This is because the means to transmit media is separate from information access, even SMS. However, it is possible to imagine some sort of assets collection being built, rather like those in moblogs (mobile photo blogs) based on photos sent by MMS or email and voice recorded by simply dialling into a recording service. It is likely that more basic phones will need to use fairly crude means to include media, but those at levels (3) or (4) should be capable of smoother interaction.

In addition to these four levels of personal mobile device, there may also be additional technology available. In particular television sets may be useful for local collaborative scenarios either using an upload and broadcast pattern (e.g. for local news slots in regional television) or using additional hardware to connect them to a phone to act as a large display. Mini-projectors on phones available in the next few years will offer further opportunities for community use, but it may be some years before these filter into rural settings.

Finally we should also note the important use of paper or other off-line materials. In a large-screen world it is easy to get into the habit of pushing all design documents onto the screen, whereas in the days of the teletypes most of the design was performed off-line on paper.

Human in the loop

Most software applications, whether on the web and on a personal computer, are focused on, as far as possible, automatically providing information or services, to the extent that they often need to include special mechanisms to make the interactions appear more personal. However, this depends critically on the cost-benefit trade-off of using humans vs. machines. In York a pilot scheme, "Net Neighbours", provided a service for elderly people to order shopping using the telephone [2]. The Internet-based shopping sites exist, but

elderly people may not know how to use the shopping web site, even if they have the requisite equipment. Instead they ring a volunteer and dictate a shopping list over the phone. The volunteer then uses the standard Internet shopping site to order a grocery delivery.

In this case the humans are volunteers, in other contexts users may be playing a game or filling in a reCaptcha code and 'doing work' as a side effect [1]. While Net Neighbours is similar to a traditional telephone helpline, reCaptcha uses people far more diffusely, sometimes called 'human computation' or 'crowd-sourcing'. When combined with a suitable payment scheme this can be a powerful means of creating micro-income possibilities that are not geographically limited and can be flexible around family and other responsibilities. One scheme, txteagle, uses SMS messages to send small tasks to those willing to spend small amounts of time in return for micropayments [21,8]. This is particular useful for jobs such as translation, which are hard to perform adequately automatically.

In a sense humans are therefore part of the technology resources available. So, in creating infrastructure for near-end-user development, we need to take into account the potential to include human computation both in the development process and as part of delivered applications.

Technology contexts

Even without human-in-the-loop computation, there is a wide range of technology contexts depending on what level of technology is available to the local developer/expert and end user, some of which are listed in Table 1. The table also lists assumed literacy levels; note that language may still be an issue even when users are literate, given the number of languages and scripts in India. We have also distinguished situations where a WAP-enabled phone may have more or less sophisticated media capabilities, however in all circumstances it is possible for the developer to create audio resources using the phone to dial into an audio content recording facility (wap+voice).

TABLE 1. Scenario contexts based on technology availability

	developer / local expert	**end-user**
A	voice+ IVR poss. illiterate	voice+ IVR illiterate
B	wap + voice literate	voice+ IVR illiterate
C	wap + voice literate	SMS/wap literate
D	wap + media literate	wap + media (il)literate
E	wap + media +community	web for the world

The first of these (A) is perhaps unnecessarily primitive as phones of reasonable complexity are widely available, but is interesting in that it offers the potential for authoring in ways that are totally language and literacy neutral. Some form of paper design is likely to be essential, but it is possible to envisage design solely using recording of voice and numeric keys for menu selection.

The last of these (E) is the situation mentioned earlier where the local community could give itself a presence in the world, either for commerce, or simply as an act of empowerment. The local users are now content producers with the local expert acting as facilitator potentially with a single higher-end phone shared by the community. This is interesting as in most cases the content provider has better technological resources than the consumer, however in this case the consumer may have a high-end browser, but the producer has only a phone. This will typically require some form of structured content as the producer will not be able to have a view of the full interface available to the final consumer. However, this works well with portal-style news sites, CMS, or blog formats.

Even apparently highly graphic interface styles can be suitable for this style of production. For example, vfridge was an early social networking site [6] (before the term was coined, it was instead referred to as a 'web sharer' application [5]). Users could attach notes and pictures to a virtual fridge door using (virtual) fridge

magnets (see Figure 1). Although the layout was graphic, the individual notes were each small and so phone based updates using WAP worked well (although the placement on the door was automatic). The technology available at the time (2000) was substantially more primitive than available in the developing world today.

FIGURE 1. vfridge - virtual Internet 'fridge door" – note small notes [6]

In the remainder of this paper we will consider two points in this design space. First a scenario of media production based on technology context (E) and the second more general programming of dynamic web content in technology context (C).

4. RICH MEDIA SHARING

End-user content production is one of the defining features of web2.0 [17]. While there are no hard boundaries, we can see that there are two main kinds of content sharing: (a) spur of the moment sharing through micro-blogging and instant moblogs (uploaded mobile camera images), and (b) slightly more crafted text and images such as more extended blog entries or YouTube videos. Of these the first demands frequent or continuous access to relatively sophisticated mobile technology, that is technical context (D) in Table 1. In contrast (b) may only require occasional use of the mobile device used for content creation and thus is compatible with technical context (E) where the local expert has access to a relatively high-end mobile phone.

Blogging software and simple content management systems (CMS) often include a relatively sophisticated rich-text editing through a web page, but also often include email portals. The latter are possible because of simple default layout and structure, typically based on the date and time of the entry. This raises the possibility for the creation and upload of media using SMS, MMS or email from a mobile device.

However, whilst this may be technologically possible, is widespread sharing of community content just another importing of western values, a sign of postmodern decadence? Surely there are more pressing needs than self-expression? In fact the opposite is the case. Those most disadvantaged financially are also often those most likely to be ignored culturally, and politically. While community and cultural historians from the late nineteenth century onwards began to look at the history of the common person (as opposed to the kings, generals and great artists), often they found the traces were missing or being lost.

The early strength of the industrial revolution has meant that this process of local culture loss, especially rural culture, was already well advanced when the pioneers of community history began their tasks. However now community history of both the distant and recent past is a major issue, for example, the first author lives on a small island of 750 people, but it has its own local history centre both for local historians and tourism. This need to root ourselves seems a universal phenomenon and is the focus of various Indian projects. Some are largely based around more traditional technology, such as the Adivasi Academy, a museum dedicated to educational programmes and preservation of tribal culture [4]; others adopt more advanced technology, such as the StoryBank project, which used mobile handsets and computers to promote digital storytelling in a small village near Bangalore [10].

As well as recalling and recording the past, end-user content is also often about recent or current events. This is the case both in the UK, for example, in the Wray Village Photo Display [19] a rural community development project developed at Lancaster University; and also in India, for example the StoryBank project was about current as much as past experience.

FIGURE 2. Wray Village display [19]

All the existing projects of which we are aware use a PC (whether individual or community) in order to organise and display content even if content capture is via mobile or other devices. So during the Winter School on Interactive Technologies bootcamp [23], we looked at how much could be achieved using the mobile device as the dominant device. This is not a detailed implementation plan, but an attempt to envisage a scenario where it is even possible and thus establish feasibility.

The capture part is relatively straightforward (see also Figure 3.A). Mobile devices capable of audio, image or video recording can always either email or MMS these. For a context where the context producers may be illiterate the MMS option may be easiest and we envisage either a single upload number or a number being allocated for each village or community (in the former case the sender's telephone number can be used for identification). This form of capture can be used also for sketches if the device is stylus or touch based, or even recognised gestures where the device has accelerometers. However, in the case of more advanced hardware, then it is likely that specialised software might be installed on the phone allowing an easier interaction than with pure MMS/email upload.

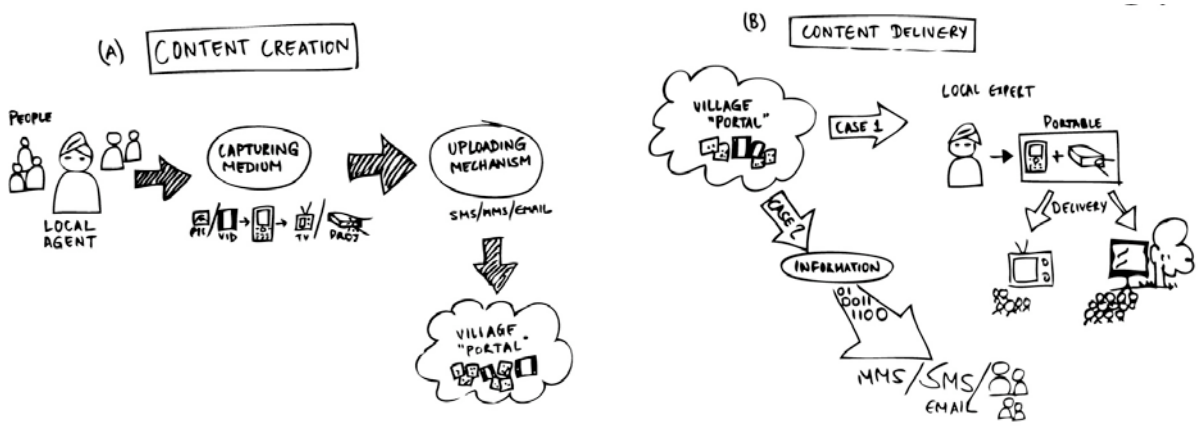

FIGURE 3. design sketches (A) content creation (B) content delivery

Staying with the case of simple MMS/email based content upload, the order of messages can be used as an (albeit slow paced) interaction dialogue allowing each interaction step to be simple. For example the upload of images to a named album (assuming a base level of literacy, but potentially entered by the local expert), would proceed as in Figure 4, with albums (content categories) being selected or created by text messages and then all rich media uploaded being added to the currently selected album.

assume Village X's unique number is 345453

1. SMS <Album name> to 345453 creates new album or selects existing one
2. Confirmation Reply delivered to the mobile phone (with unique content id)
3. Send MMS of video to 345453 upload content to the album
4. All further content from this phone goes to the selected/created album

FIGURE 4. design scenario for SMS/ MMS upload

Note that in the case of illiterate end-users the selection of the current album/category could be based on simply being able to enter text characters by rote, or the album selection may be done by the local expert. As an alternative, album selection could be achieved by photographing an icon or glyph and then using server-end image recognition as has been used in a number of location-based services [18]

Note that the use of SMS/MMS allows actual content upload by village people (end users) using relatively low-end camera phones (technology context C) as well as by the local agent/expert with a more advanced mobile/smart phone. The content that the people generate could be photos or videos taken during festivals or functions that happen in the village, or may relate to agriculture or health.

Content generated in this way could be automatically published on websites for viewing globally or delivered back to mobile devices via MMS or WAP portals. However, for the community as a community more group-based presentation is possibly better than the individual views of mobile devices. For example, the *Namma Dhwani* community-broadcasting initiative used a combination of distribution means from cable television and loudspeakers in public areas (as described in [10]). It is common for television to be used by small communities or extended families and the potential of using the existing television infrastructure has been explored as a means to distribute material for printing [22]. Similarly adding a small piece of hardware, a form of set-top box 'or video-out adapter', could allow the local-expert's mobile device to use the television as a larger output device, probably with no more resolution than the mobile, but simply larger size for group viewing. Several high-end media phones already have TV-out as a built-in feature, so for these no additional hardware would be necessary. As the emerging mobile phones with micro-projectors become commoditised, this will become an alternative means of achieving the same goal.

5. TINY-SCREEN CODING

While content creation maybe possible through a small-screen device, the authoring of more complex hypermedia and certainly programming seem much more difficult. For example Eclipse, the popular open-source IDE, looks cramped even on a laptop screen (Fig. 5.A) and ceases to be usable on phone (Fig. 5.B).

FIGURE 5. Eclipse (A) full laptop screen (B) phone-sized window

However, this apparent need for large screen displays is perhaps more a feature of availability than necessity. Early programming was done using punch-cards and many of today's programmers cut their teeth on home-computers of the 1970s using TV screens as 40x25 character displays, not significantly different from that available on many smartphones. The differences are partly about coding culture and partly that language design itself reflects the assumed technology and audience.

One feature of 'modern coding' is that many programmers work entirely on-screen whereas in the past the paper listing dominated coding practice (see also [7]). In Marsden et al.'s work, the local expert (human access point) used a PC-based graphical environment to create questionnaires [13]. However, one could equally imagine this graphical design being performed on paper with the elements being given names or codes as they are created online and some form of fairly arcane linking notation. This may not be the ideal production environment, but would at least make it possible to produce content with minimal infrastructure (and anyway many recent notations such as XSLT are hardly designed for ease of use!). However, as multi-touch devices become more commonplace, these may make it easier to manipulate more graphical representations directly.

Similarly the 1970s home computer user would write code (usually in Basic) using hand-written listings. The line-number focused editing with 'gappy' stable line numbers (rather than serial line numbers starting at 1) facilitated this combination of paper-based and on-screen coding. As a design principle:

Lesson 1. design environments and languages to support off-line representations

These early programming systems were not only limited in terms of screen space, but also computational power and memory. Current phones are now 100 times faster and have 1000 times as much memory as these home computers, and the back-end web systems they connect to are orders of magnitude faster and larger again. So, while early small-screen computing shows that phone-based coding is a reasonable expectation, we would expect that the trade-off in language and programming environment features will be different now.

One feature that was common to home computers and also some of the most innovative programming environments, was the close relationship between programming language and environment. For example in Smalltalk the two are hardly distinguishable. Furthermore development environments dedicated to user interface development such as HyperCard and Visual Basic (albeit much maligned by computer scientists) tend to elide the difference between design and execution, to the extent that design changes are often possible while the system 'runs'. In all of these cases, the 'code' is not a monolithic sequential file, but broken into parts that are linked by the system. All of these run counter to the more traditional design of programming languages as things to be written in files. Design environments and debuggers are 'add-ons' to the basic language.

For small-screen devices these seem to be important issues. If the environment can do more of the work, then this makes sense. For example, in classic programming languages names of variables, functions and methods need to be completely unambiguous leading to long expressions such as `'record.personnal_details.address.street_number'`; hardly conducive to small screen entry! However, if code entry and listing are seen as online user interaction, then it is sufficient that names can be disambiguated (potentially interactively) during code entry and are clear to the user during browsing, for example, '`street_number`' may be sufficient if there is only one variable with the named field in scope, and users could be allowed aliases such as "strnum' that expand on demand. The *internal* representation can be as detailed and unambiguous as necessary. Similarly, scripting languages often allow variables without explicit type definitions or declarations, but production languages demand more rigorous, but verbose pre-declaration. There is no reason why the environment should not effectively create inferred types and detect potential mis-spellings as it interprets small snippets of entered code (not large files). More broadly we may be able to exchange interaction for syntax.

Lesson 2: the environment is part of the language

Another example of this is Knuth's Literate Programming [11,12]. This was based on traditional languages (Pascal in his original WEB, but later also C and other languages), but effectively created a new way of writing code that was not governed by the sequence order of the programming language and freely mixed code and documentation. A key feature was the ability to replace whole sections of code that can be given a name, so allowing more succinct views, rather like pseudo-code.

```
if ( << input invalid >> )
<< error handling code >>
else
<< update data >>
```

FIGURE 6. use of named code snippets in literate programming

This hypertext-like coding structure allows one to understand the 'gist' of the code, especially important on a small screen.

The elision of design and use is also potentially powerful for tiny-screen coding. User-interface design is expected to be an iterative process; so we expect to run code, see problems in the behaviour and then fix them. It can be hard to trace the code that creates a behaviour even with a large screen, and so this would be doubly difficult on a mobile-phone screen. However, environments such as HyperCard, which elide design and use, mean that when a behaviour issue is identified the system can immediately be switched into development mode and one is effectively at the right point in the 'code' to fix the problem. Typically the 'code' in these environments is an event handler and relatively small, again making it well-suited to tiny-screen coding.

Lesson 3: reduce the gap between design and execution

This is fine if the designer and coder are the same person. If not then the same general principle can be used. If during the use of a mobile application the designer (who is not the coder) notices an error, or potential improvement, then why not edit the output and effectively create both a specification for the coder to follow and a test case that can be automatically checked, as is promoted by agile development practice.

In general, bridging the gaps between environment and language, design and use, test and bug report tend to allow greater localisation, important on a small screen, and also are features found in many end-user or near-use software such as spreadsheets (eliding data, code and execution), Yahoo! Pipes (design close to execution), and programming by example (use is design).

These types of features also are well suited to more use-case oriented development, which tends to follow linear scenarios of use, again easier to keep track of on a small screen. Traditional programming languages often require kludges for scenario-based coding, such as leaving blocks of code empty that we know will later be needed, but with the danger that these might be forgotten. However, adopting a literate programming style of coding it would be reasonable to leave a section such as '`<<error handling code>>`' unwritten. Now it is explicit in the code in Figure 6 that there is missing code, and so the system can add a 'to do' for the coder. However, there is no reason why this should not be executed so long as the first branch of the 'if' statement is not needed. However, if the branch is required, then instead of simply stopping, a system that elides design and use could prompt the coder then and there to either supply the missing code (just-in-time-code) or to emulate the effects of the code (thus generating a test case). Potentially this could even extend to releasing incomplete code (it happens anyway!) and then using human-in-the-loop techniques to supply results when needed.

6. CONCLUSION AND FURTHER WORK

This paper is an exercise in conceptual feasibility, not a finished work. We hope it demonstrates that tiny-screen content production and even more complex scripting or coding is possible. The next stage is to produce proof-of-concept prototypes. In fact, there are a number of existing tools available or in the pipeline. For plain coding, Codepad, a web-based tool for executing small snippets of code, says it already works well on a mobile [9], but this is for short code snippets, not larger code such as a user interface or web application. Mozilla are also about to release a web-based code editor; while it is clearly aimed at the big-screen web, this suggests that a similar web-based environment for mobile devices may be possible. Perhaps most promising is that Ken Banks and Gary Marsden who suggested the discussion topic at the Mobile Design Dialogue, have since become some of the partners in a project 'mobility' looking at mobile-based programming tools [16], although this has only started work recently and has not yet produced any outputs.

REFERENCES

[1] von Ahn, L., Maurer, B., McMillen, C., Abraham, D. and Blum, M. (2008) reCAPTCHA: Human-Based Character Recognition via Web Security Measures. In *Science*, Vol 321, 12 Sept. 2008. pp. 1465–1468

[2] Beardon, H., Munyampeta, F., Rout S. and Maiso Williams, G. (2005) *ICT for Development, Empowerment or Exploitation: Learning from the Reflect ICTs project*. ActionAid. http://www.actionaid.org.uk/1413/ict_for_development_empowerment_or_exploitation.html

[3] Blythe, M. and Monk, A. (2005) Net Neighbours: adapting HCI methods to cross the digital divide. Interacting with Computers. *Interacting with Computers*, 17, 35-56.

[4] Coates, B. and Coates, E. *Museum of Voice*. International Workshop: Re-Thinking Technology in Museums: Towards a New Understanding of People's Experience in Museums. Limerick (Ireland), 29th-30th June 2005

[5] Dix, A. (1999). *The Web Sharer Vision*. eBulletin. aQtive ltd. November 1999. http://www.hiraeth.com/alan/ebulletin/websharer/

[6] Dix, A. (2001) Designing a virtual fridge. *Computers and Fun 3*, York, 13th December 2000. (appears in Interfaces vol 46 (Spring 2001), pp.10-11)

[7] Dix, A. (2008). *As We May Code - The art (and craft) of computer programming in the 21st century*. keynote at PPIG08 – The 20th Annual Psychology of Programming Interest Group Conference. Lancaster University, UK. 10–12 September 2008.

[8] Eagle, N. "txteagle: Mobile Crowdsourcing". To appear in *HCII '09*.

[9] Hazel, S. (2009) Codepad. (accessed Sept. 2009) http://codepad.org/about

[10] Jones, M., Harwood, W., Bainbridge, D., Buchanan, G., Frohlich, D., Rachovides, D., Frank, M., and Lalmas, M. (2008) *"Narrowcast yourself": designing for community storytelling in a rural Indian context.* In Proceedings of the 7th ACM Conference on Designing interactive Systems (Cape Town, South Africa, February 25 - 27, 2008). DIS '08. ACM, New York, NY, 369-378. DOI= http://doi.acm.org/10.1145/1394445.1394485

[11] Knuth, D. (1992) *Literate Programming* Stanford, California: Center for the Study of Language and Information, 1992, (CSLI Lecture Notes, no. 27.) ISBN 0-937073-80-6 http://www-cs-faculty.stanford.edu/~knuth/lp.html

[12] *Literate Programming*. (accesed July 2009) http://www.literateprogramming.com/
[13] Marsden, G., Maunder, A. and Parker, M. (2008) People are people, but technology is not technology. *Phil. Trans. R. Soc. A* (2008) 366, 3795–3804. doi:10.1098/rsta.2008.0119. http://people.cs.uct.ac.za/~gaz/papers/RSTA20080119.pdf
[14] Mitra, S., Dangwal, R., Chatterjee, S., Jha, S., Bisht, R. and Kapur, P. (2005) Acquisition of computing literacy on shared public computers: Children and the "hole in the wall" *Australasian Journal of Educational Technology*, 2005, 21(3), 407-426. http://www.ascilite.org.au/ajet/ajet21/mitra.html
[15] *Mobile Design Dialog*. Cambridge. 3–4 April 2008,
webpage: http://www.cs.swan.ac.uk/mobdesign/
Mobile Design Dialog discussion: http://mobiledesigndialog.nexo.com/
[16] *mobility*. (accessed July 2009) http://mobility.kiwanja.net/
[17] O'Reilly, T. (2007) *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, O'Reilly Media, 30th Sept. 2005, (accessed 23/06/2007) http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
[18] Stewart, J, M. Wright, H. Ekreus, R. Coyne and P. Travlou (2009) *The Memory Space – Exploring Future Uses Of Web2.0 And Mobile Internet Though Design Interventions*, COST 298 conference, Copenhagen, May 2009
[19] Taylor, N. and Cheverst, K. and Dix, A. and Silva, P. and Rouncefield, M. (2008) The Co-realisation of a Village Photo Display. In: *CHI 08 Workshop on Collocated Social Practices Surrounding Photographs*, 05 Apr 2008, Florence, Italy. http://eprints.comp.lancs.ac.uk/1437/
[20] Tobin, L. (2009) Slumdog professor: The academic whose work inspired the Oscar-winning film has a new project in the UK. *The Guardian*, Tuesday 3 March 2009
[21] *txteagle: Empowering the largest knowledge workforce on Earth*. (accessed July 2009) http://txteagle.com/
[22] Vennelakanti, R. and Gupta, A. (2009) *Leveraging Existing Television Infrastructure - Enabling Broadcasting and Printing Documents via TVPrintCast*, Workshop on Human-Centered Computing in International Development, CHI '09, April 4-9, 2009.
[23] *Winter School on Interactive Technologies*. UK-India Network on Interactive Technologies. (HP Labs in Bangalore, 2nd & 3rd February 2009). http://www.ukinit.org/02122008/winter-school-interactive-technologies