Opening the Box
Meta-level Interfaces Needs and Solutions

Alan Dix
Talis, 43 Temple Row, Birmingham, B2 5LS, UK
and Birmingham University, Edgbaston, Birmingham, UK
alan@hcibook.com
http://www.hcibook.com/alan/papers/SUI2011-meta/

ABSTRACT
This paper begins by considering reasons why some form of meta-level interface may be required for modifying or exploring existing user interfaces, from obvious functional reasons of customisation and personalisation to more political and social goals such as education and empowerment. The paper considers examples of systems developed by the author and others, and uses these to present a number of techniques and principles for effective meta-interactions. Some of these concern more surface manipulation, and others deeper levels of code and meta-descriptions of the application and UI. It concludes that meta-interaction may be a key element for future liberal society.

Keywords
customisation, personalisation, end-user programming, end-user empowerment, appropriation

1. INTRODUCTION
The topic of this workshop brings together a number of areas on which I have worked or that have been of personal concern. This paper will discuss some of these areas of concern and then look at general principles and techniques that can be used to address them.

2. WHY META?
While it hardly needs stating for this workshop, to many it may seem that meta-level interactions are simply the preserve of the hobbyist or techie. However, they are both ubiquitous and of broad benefit.

2.1 Customisation and Personalisation
Of course meta-level user interfaces are common. Every time a user drags a palette to the side of the screen, selects a ringtone or modifies the style definition in a document, she is engaging in an adaptation of the user interface. However, we also know that beyond a few examples like this few users actually customise despite having problems or gripes that could be dealing with through simple selection of options (for example, turning off some of the 'smart' features in Word). Improving even these basic features can have a major impact on user experience.

2.2 Appropriation
In particular "plugability and configuration" is one of the design principles for appropriation [9]. Indeed several of the design principles discussed in [9] are related to meta-level user interactions; while appropriation is possible using the interface as given, the user has greater flexibility if she can peek under the hood (design principle "provide visibility") and tinker inside ("plugability and configuration") and share the results with others ("encourage sharing").

2.3 End-user Empowerment
One advantage of appropriation is the sense of ownership and empowerment it engenders. A sense of control is important for well being, and the act of tinkering gives this, whether to improve the user interface for its original purposes, or make it do something completely novel.

While this is important for all users it is particularly relevant for those in developing countries, or the disadvantaged in developed countries, who can be doubly disadvantaged in a world where access to information is central to economic and political power [1].

Existing technology can be appropriated by traditionally disadvantaged groups; for example, Jensen reports how mobile phones allowed fishing boats in Kerala, southwest India, to obtain higher prices for their catches [12] and we have all seen the impact of social media in recent popular uprisings across North Africa and the Middle East.

However, if those closer to need are in a position to create, modify or adapt existing software and hardware the results are likely to be more appropriate than tools designed primarily for an urban, middle-class, western environment. This may be the end user, but Marsden et al. argue the case for 'human access points', local experts, in their case local health workers, who are given the tools to create and adapt mobile-phone administered questionnaires [16]. Prompted by various workshop discussions [17, 20], we have explored the potential for a range of mobile phone-based adaptations including compete coding via the mobile-phone screen [10].

2.4 Education

Often modifications to user interfaces require a high degree of expertise; so education is needed in order to use them. However, if well designed, meta-level interactions hold the potential to be a means for education in themselves; as generations of children who have fiddled with old car engines can testify. Education, of course, also contributes to empowerment.

The Query-by-Browsing (QbB) intelligent database interface is an example of this. QbB generates SQL: queries based on user record preferences, but then reflects this back to the user both by highlighting the records selected by the query and by exposing the query itself [7]. The user can comprehend the system via the concrete record selections, but in the process learn the SQL that produce it (although not the machine learning algorithms which generate the queries).

2.5 Privacy and Auditability

The control of privacy settings in social applications such as Facebook, has become a big issue. Höök also argues that this is an issue likely to be important in future ubiquitous computing applications [11]. Indeed the very openness in low-level architecture required for rich context-sensitive features in itself creates privacy issues [8]. Many approaches to privacy, in ubiquitous computing and elsewhere, focus on restricting information flow. However I have long argued that it is the eventual *use* of the information that is most critical [6]; that is systems that expose what happens to information both currently (visibility) and in the past (auditability) are far more likely to support the user's ability to manage information disclosure.

2.6. Comprehensible Behaviour and Trust

Closely related is the issue of trust, not just for financial and or personal security, but also at a mundane level of whether we decide to use particular application features. This is especially important when systems make choices automatically for us. The kind of openness needed to allow a user to adapt a system is very similar to that needed to allow a user to believe in what it is doing already.

The record listings in Query-by-Browsing [7] are an example of this as they may be comprehensible to the user, even if the SQL is not, giving the user confidence that the query will continue to be appropriate for unseen records. Another example is MICA, which makes suggestions for GUI customisation based on user activity, but also "*includes a description of why MICA is making recommendations and how it generated them*" [5], precisely to support Hook's "predictability and transparency" principle [11] and so engender trust.

3. TECHNIQUES AND PRINCIPLES

So if meta-level investigation and modification is a good thing, how can it be achieved?

3.1 Cost and Benefit – When it happens

Sometimes people don't customise because they don't know how. However many experts do not customise their interfaces even if they complain about the things that are wrong! The key problem is not lack of understanding but lack of immediate benefit. We are creatures who heavily discount the future; effort now for future gain is hard. If customisation can be made closer to the point of use it becomes more likely. One example are dialogues that ask for a decision, but have a tick box to say "always do this". This is effectively asking you set a preference, but at a point in time when you are in the middle of doing the requisite action. The benefit is clear and the cost (in terms of clicks and mental effort) low. Furthermore this is all set within the context of a concrete example of use (see also next point)

3.2 Progressive Disclosure –Where It happens

The preferences and customisation of many applications are buried in a "preferences" menu item far away from the actual interaction. Somewhere in a preferences panel you set parameters whilst guessing vaguely what they might be about. However, others connect customisation closer to the thing it affects. Back in 1995, Marsden [15] advocated the advantages of a systematic policy suggesting a 'screw' metaphor where every component has a small screw icon in the bottom right hand corner. Clicking the screw 'undoes it' revealing the circuitry within, and potential the ability to unscrew other sub-components (see Figure 1).

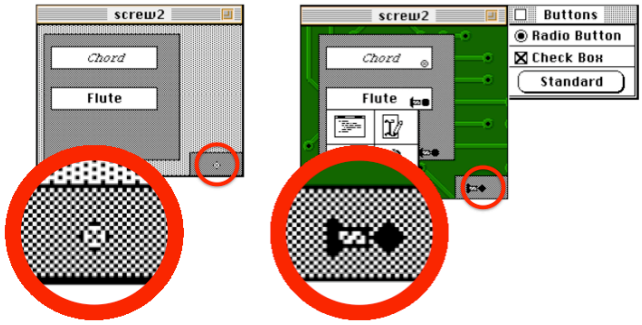

Figure 1. Screw Metaphor from [15]
(a) screw in – UI (b) screw out – metaUI

Today in the Apple Dashboard just such a mechanism is found on widgets. Instead of a screw a little 'i' for information icon, clicking it 'turns around' the widget showing settings behind. Strangely the iPhone reverted to a special place for settings rather than associating them closely with their application.

Figure 2. Mac OS Dashboard widget
(a) front – UI (b) back– metaUI

3.3 Tools of Revelation

A similar approach is to use some form of external 'tool' for meta-level modifications. This happens in the real world; Figure 3 shows a stud detector, which detects the wooden studs in a wall so that you can screw into them. The wooden structure is hidden behind plasterboard and wallpaper, but the stud detector reveals it – the "provide visibility" appropriation principle [9] in the physical world.

Figure 3. Wall Stud Detector

Note that "provide visibility" does not mean the same as Nielsen's "visibility of system status" evaluation heuristic [19], as this usually refers to the essential information about the system for normal use. Instead, if systems reveal a little more (such as a mobile phone showing signal strength not just whether or not a call can be made), then the user can use this in unexpected ways (such as waving the phone about to seek out better signal).

Beaudouin-Lafon's 'instrumental interaction' [2] and in particular Toolglasses [3], follows the same principle as the stud detector advocating the use of 'instruments' as a means for modifying and interacting with objects.

3.4 Smooth Transitions

When creating means for user to modify their environment there is often a temptation to try to do everything – the spectre of Turing equivalence rises and before long a simple end-user customisation tool becomes a full-blown and complex programming language. The effort to produce something that could, *in principle*, do everything often ends up with something that, *in practice*, is good for nothing. However, the alternative is often to have very different means for simple and more complex modifications, so that users hit barriers; for example, moving from Excel formulae to Visual Basic.

Mathematicians face a similar problem when modelling 'differential manifolds' curved spaces such as the surface of the Earth or the curved space-time of general relativity. They effectively paper the curved space with flat Euclidean surfaces (which are easier for a mathematician to handle), but if you try to use a single flat surface there is at least one point where things go very wrong, like the place where the foil is all folded up at the end of an Easter egg. Instead mathematicians use a collection of small patches, which overlap in a 'smooth' manner.

One can envisage customisation working like this, with different levels of customisation (perhaps ending up at open-source code), where the two ends (use and coding) have a huge gulf between them, but where each pair of successive levels overlap with an easy transition. This sounds like a hard problem, but there are examples that achieve this to varying extents. HyperCard had a smooth transition from use to customisation and then to programming. In consequence, many who would never consider themselves programmers created complex HyperCard applications. Xerox Buttons were another example, where a non-technical user might just use the button, then peek at its code and change a file name, and perhaps, over time, start to understand some of the code that drove the familiar user-interface actions [14]. Could the Excel formula to VB step be more like this?

3.5 Ease of collaboration

Another of the appropriation principles is "encourage sharing" [9]. In Nardi and Miller's classic study of spreadsheet use [18], they describe the collaboration between Buzz and Betty

> *"When Buzz helps Betty with a complex part of the spreadsheet such as graphing or a complex formula, his work is expressed in terms of Betty's original work. He adds small, more advanced pieces of code to Betty's basic spreadsheet: Betty is the main developer and he plays an adjunct role as consultant."*

The fact that spreadsheets have relatively smooth transitions (at least between levels of formula use) make this collaboration possible. Note especially that Betty is able to do a lot herself, and probably extends this over time (education). Furthermore Betty is able to determine her own level and understand when to seek help.

Spreadsheets, by their nature allow them to be passed around. It is far rarer to see other kinds of configurations shared. In UNIX systems, a lot of configuration is in text files, such as .login or .profile, and expert users will move these around. However, it is near impossible to simply take one person's Word settings and apply them to another users machine. Xerox Buttons [14] were a simple idea, a button that executed some Lisp code, but were surprisingly powerful, in part because you could *mail them round*, creating a community. Maker cultures emerge when people can share ideas and, even better, artefacts.

3.6 From Configuration to Code

Spreadsheets, Xerox Buttons, Query-by-Browsing and HyperCard are all examples where the user can move in steps from doing things to raw coding. When looking at near-end-use development, one of the design lessons was "reduce the gap between design and execution" [10].

> "*In general, bridging the gaps between environment and language, design and use, test and bug report [...] 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)*"

At Talis we are working on tools to bridge this gap for linked open data [4] as exposed, for example, in

data.gov.uk. This is building on Callimacus, where RDFa embedded in a web page turns it into a UI generation template, opening up application building to ordinary web developers [13].

3.7. Meta-Representations for Meta UIs

As well as being the subject of user interaction, semantic data of some form seems to be a key element of future user interactions. Whether mashing data for the web or connecting digital devices in the living room, effective meta data about devices, applications and their interactive potential seems an essential start point for more flexible machine initiated activity, for machine activity to be explicable, and for users to be able to interrogate and modify it. Model-based user interfaces are clearly one way to achieve this, but there could be other solutions, similar to the way applications expose meta-information for Apple Scripting on Mac OS or via COM on Windows.

4. CONCLUSIONS

We have discussed various principles and methods for meta-level interactions., and also some of the reasons why this is 'a good thing'. As we enter an era of open data and mashups the ability to digitally tinker seems not just a hobby, but a key enabler of a broad-based civil society.

REFERENCES

1. Beardon, H., Munyampeta, F., Rout S. and Maiso Williams, G. ICT for Development, Empowerment or Exploitation: Learning from the Reflect ICTs project. ActionAid. (2005) http://www.actionaid.org.uk/1413/ict_for_development_empowerment_or_exploitation.html
2. Beaudouin-Lafon, M. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. in *Proc. of the CHI '00.* ACM Press, (2000) 446–453.
3. Beaudouin-Lafon, M and Mackay, W. Reification, polymorphism and reuse: three principles for designing visual interfaces. in *Proc. of AVI '0)*. (2000) ACM Press ,102-109.
4. Bizer, C., Heath, T. and Berners-Lee, T. Linked Data – The Story So Far. *International. Journal on Semantic Web and Information Systems*, 2009.
5. Bunt, A., Conati, C., and McGrenere, J. Supporting interface customization using a mixed-initiative approach. In *Proc. of IUI '07*. (2007) ACM Press. 92-101.
6. [Di90] Dix, A. (1990). Information processing, context and privacy. in *Proc. of INTERACT'90*, (1990) North-Holland. 15–20. http://hcibook.com/papers/int90/
7. [DP94] Dix, A. and Patrick, A. (1994). Query By Browsing. Proc. of IDS'94: *The 2nd International Workshop on User Interfaces to Databases*, (Lancaster, UK, 1994), Springer Verlag. 236–248. http://hcibook.com/alan/papers/QbB-IDS94/
8. Dix, A. Beyond intention – pushing boundaries with incidental interaction. Proc. of *Building Bridges: Interdisciplinary Context-Sensitive Computing*, (Glasgow University, 9 Sept 2002) http://hcibook.com/alan/papers/beyond-intention-2002/
9. Dix, A. 2007. Designing for appropriation. In *Proc. of HCI2007 Volume 2*. (2007). BCS, 27–30. http://www.bcs.org/content/conWebDoc/13347
10. Dix, A., Kozhissery, R., Ravichandran, R. and Dayanand, D. Content Development Through the Keyhole. in *Proc. of EISE2009, Expressive Interaction for Sustainability and Empowerment*, (2009) 67–78. http://hcibook.com/alan/papers/EISE2009-Keyhole/
11. Höök, K. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12 (2000) 409--426,.
12. Jensen, R. The Digital Provide: Information (Technology), Market Performance, and Welfare in the South Indian Fisheries Sector, *Quarterly Journal of Economics*, 122(3), (2007) 879–924.
13. Leigh, J. and Wood, D. RDFa as a Query Language. *Semantic Technology Conference*. (June 2010)
14. MacLean, A., Carter, K., Lövstrand, L., and Moran, T.. User-tailorable systems: pressing the issues with buttons. In *Proc. CHI '90.* (1990) ACM Press, 1990, 175–182
15. Marsden, G. Overcoming Design and Execute Modes in User Interface Design Environments. in *Proc. of HCI 95 people and Computers* (1995), 133-137
16. Marsden, G., Maunder, A. and Parker, M. People are people, but technology is not technology. *Phil. Trans. R. Soc. A* (2008) 366, 3795–3804.
17. *Mobile Design Dialog*. (Cambridge. 3–4 April 2008) webpage: http://www.cs.swan.ac.uk/mobdesign/ Mobile Design Dialog discussion: http://mobiledesigndialog.nexo.com/
18. Nardi, B.and Miller, J. An ethnographic study of distributed problem solving in spreadsheet development. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work (CSCW '90).* (1990) ACM, Press, 197–208.
19. Nielsen, J. Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection Methods*, (1994) John Wiley & Sons, New York, USA.
20. *Winter School on Interactive Technologies*. (HP Labs in Bangalore, 2nd & 3rd February 2009). UK-India Network on Interactive Technologies. http://www.ukinit.org/02122008/winter-school-interactivetechnologies