Towards Explainable User Interfaces

Alan Dix
Cardiff Metropolitan University
and Swansea University, Wales, UK
alan@hcibook.com

Tommaso Turchi
Department of Computer Science,
University of Pisa, Pisa, Italy
tommaso.turchi@unipi.it

Ben Wilson
Computational Foundry,
Swansea University, Wales, UK
b.j.m.wilson@swansea.ac.uk

This paper argues that user interfaces need to be explainable whether or not they contain artificial intelligence components. Even with the best design, complex applications often leave users confused; this is exacerbated on small touchscreens, where small slips can lead to markedly different outcomes and when notifications or intelligent agents may autonomously change the interface. This can be disorienting even for the most tech savvy user, but doubly so for those less confident or with motor-control issues. We are often left asking "what just happened?" or "how can I do this again?". We need explainable user interfaces.

User experience, accessibility, artificial intelligence, notifications, user interface architecture

1. THE PROBLEM

"What just happened?!" We have all experienced this; you are in the middle of doing something and the screen suddenly switches apparently randomly. If you are experienced you think, "*I must have touched something by accident*" or, "*perhaps the page timed out*", maybe you try the back button on a web page or on an Android phone, or simply back-off and try again. Sometimes this is just disorienting and mildly annoying, sometimes you lost the text you spent half-an-hour carefully crafting. Now imagine you are less confident, perhaps older or less experienced, or have motor control issues either through illness, disability or due to an activity such as running or driving or being on a bus.

Things may go well: you have just managed to create a new payee on your banking app, and you wonder, "*how did I do that?*". Or you've been in the middle of something, got interrupted by a notification or phone call and then think "*where was I?*".

AI can make this worse acting autonomously or because a side effect of making the interpretation of our actions more intelligent is that they are often less predictable and harder to make sense of. For AI we are used to thinking about issues of explainability (Guidotti et al. 2018, Hassija et al. 2024); indeed, the author has been writing about the importance of transparency to ameliorate machine-learning bias since the early 1990s (Dix 1992), and the 'right to an explanation' is enshrined within EU law (EU 2024).

However, AI is simply adding to existing UI complexity. With or without AI, we need to be able to ask "*what happened?*", "*what did I just do?*", "*how can I do this again next time?*".

We need explainable user interfaces.

This is a challenge to the HCI community that has impact on interaction paradigms, design heuristics, implementation architectures and tool support.

In the rest of this paper we'll look at the pressures that have made explanations more necessary, potential ideas on what XUI designs might be like, and architectural implications.

2. WHY IS EXPLANATION NEEDED?

Shouldn't good design mean that explanation is never necessary? Predictability, error prevention, visibility and feedback have been key UI design principles since the early years (Dix et al. 2004). Yet, despite more than forty years of design advice, these problems seem to be getting worse. This is not simply designers ignoring advice, but a symptom of long-term trends and changes in user interfaces.

First there is a shift in the *locus of control*. Early models of interaction, including Norman's (1998) seven stage execution–evaluation cycle assume that the usual mode of interaction is the user issuing some sort of command or instruction and the system responding. However, even in early days collaborative user interfaces, systems controlling physical systems or responding to external events, or simply timing led to apparently non-deterministic behaviour and race conditions. Notification-based systems have increased this trend, and the likelihood of greater numbers of AI agents acting on our behalf exacerbates this further.

There are changes in *complexity and scale*. AI and big data mean that systems are harder to comprehend and the shift to mobile means that less state is visible, and so there are often more steps, screens or modes in each interaction.

Finally, the drive towards efficient interactions often leads to *shrinking gaps* between user actions. Saussure claimed that language evolves to create ‘binary opposition’, clear distinctions to enable unambiguous communication (Fogarty 2005); similarly Goodman (1976) describes how the discretisation of musical notation enables us to be able to say “*this is Beethoven’s Fifth*” not simply improvisation inspired by it. In interfaces these distinctions are blurring *spatially* (fat finger, two vs three finger drag), *temporally* (length of presses, changing meanings and appearing/disappearing targets) and *semantically* (spelling corrections, autocomplete, speech recognition). Tiny differences in our actions give rise to very different results; that is *slips* are easier (Reason 1990).

3. HOW CAN EXPLANATIONS HELP?

Explainable interfaces help users in several ways, offering *understanding, control and empowerment* to those who feel constantly confused. First, an XUI is important when things go wrong to avoid users blaming themselves or feeling stupid. In such circumstances it also helps users’ *undo or repair*. When government and commerce is digital-first this is not simply a matter of individual benefits but is essential to ensure an active and engaged citizenry.

Second is to help users *replay/redo* things that did work. Having spent time finding a successful sequence of menu sections, swipes and/or obscure command key combinations, you want to be able to do it again, but more reliably. Weirdly, although ‘recognition rather than recall’ was a motto of the move from CLI to GUI, the need to remember ever more has become a feature of recent interfaces. It is not uncommon to see older (and younger) users with notebooks full of step-by-step instructions on how to use an application – explanations could be editable and/or automatable, or simply make it easier to create personal “how to” guides.

Finally, XUIs make it easier to *provide help*. A third-party helper, if told, “I was in the middle of paying my bill, pressed something and I got here”, could go to the system and find the sequence of actions that led there. Furthermore, when things go right, the user can share their knowledge more easily with others.

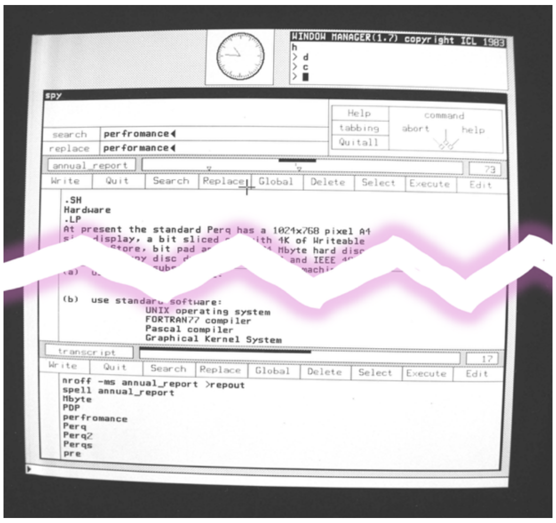
3. WHAT XUI MIGHT BE LIKE

Explainable interfaces might adopt several styles, just as there are many different kinds of human explanations for different purposes.

3.1 Commands and scripts

In previous work the author proposed ‘nasal interfaces’, not in the sense of literally using the nose, but like smell having a sense of history (Dix 2022) unlike visual interfaces, which are all about the instantaneous state of the system now. The interface providing a sense of history was precisely in the context of helping elderly or other users understand what has been happening. Command line interfaces (CLI) naturally have a history of past interactions, but scripting also has this form, for example Apple Script automation (Cook 2007), as do chat-based interactions.

There are good reasons why the DOS or UNIX command line, has been largely replaced with GUIs! However, there are and have been systems that have combined CLI-style with graphical interface features, which might offer inspiration.



```
h
> d
> c
> █
```

```
.SH
Hardware
.LP
At present the standard Perq has a 1024x768 pixel A4
si  display, a bit sliced    ith 4K of Writeable
    Store, bit pad ar       Mbyte hard disc
      ppy disc d          and IEEE 4
           subs              machi
(a)  u

(b)  use standard software:
             UNIX operating system
             FORTRAN77 compiler
             Pascal compiler
             Graphical Kernel System
```

```
nroff -ms annual_report >repout
spell annual_report
Mbyte
PDP
perfromance
Perq
Perq2
Perqs
pre
```

Figure 1. RAL's SPY Editor – Note UNIX shell in bottom pane. (Source: Chilton 2022, section 36; © UKRI Science and Technology Facilities Council, 1983)

In the early 1980s, Rutherford Appleton Laboratories created Spy, a programmer’s text editor for the new ICL PERQ workstation (Chilton 2022, Section 40.5). The editor used vertical pans to fit with the portrait A4 display. Any pane could be designated to include a UNIX shell (see lowest pane in Figure 1). All text editing functions operated on this pane, so it was possible to copy outputs and also edit past shell commands and then copy them to the end of the window to re-execute. Furthermore, the transcript could be saved and reopened, just like any other pane. The author’s habit was to have a file, “doit.txt”, in each directory holding the common commands for processing in that directory (e.g. coding, text preparation using Troff) which would build up over time, sometimes being ‘tidied up’, or even extracting commonly used sequences as a shell script.

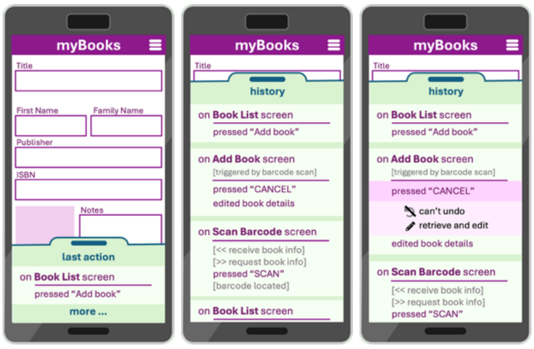


Figure 2. Application with history – envisionment

The modern equivalent of this are notebook-style interfaces, notably Jupyter Notebook, which acts like a python CLI with interspersed outputs and allows past input to be edited (Kluyver et al. 2016). Note that in this notebook-style of interaction (and the Spy editor), edits to the past commands are not themselves tracked as history (except for undo), so there can still be 'what did I do' moments, although some variants also include versioning.

Versioning systems include branching, but this is heavy weight and in general there is little support for true trial-and-error interaction styles in user interfaces. In the early days of undo systems there were some very arcane proposals which were too complex for the time, both in terms of computational and (pre-bitmap!) screen interaction (Archer et al. 1884). However now with richer interaction possibilities and near billion-times faster computation and storage, it may be time to revisit branching histories allowing one to step back to previous states and try alternatives.

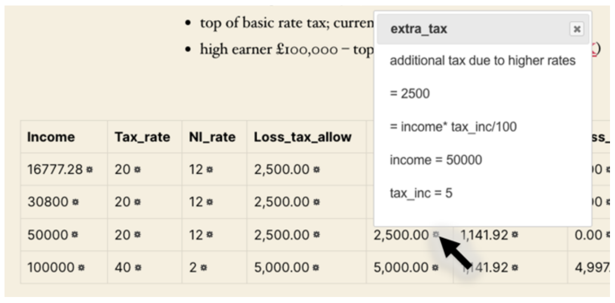
3.2 What happened when – scripts and undo

The simplest form of explanation is *temporal*, a step-by-step recap of what happened (see Fig.2). While this comes 'for free' in a scripting interface, it can also be provided where the immediate interactions are graphical. At one stage Apple encouraged applications to be factored using Script-level events between the UI and back end, a simplified variant of the Seeheim architecture (Pfaff 1985).

In fact, many graphical systems do have the information for this as part of undo support, indeed some have an undo menu listing past actions to enable multi-step undo. Where this is not explicit to the user, some users perform undo/redo and watch for the change in order to address "*what just happened?*"; a form of appropriation.

Scripting and undo systems usually have a fixed granularity, but to be a good explanation, this step-by-step may need some form of hierarchical *structure*: at the highest level a user action such as a button press or event such as message arriving, with lower levels tracking internal events that have been triggered by these top-level items.

Even a single temporal event may need to be described at different levels of *abstraction*: lexical: the user touched the screen; syntactic: the "save button" was pressed; semantic: request to save document. AppleScript allows actions at each of these levels, but, as its intention is to help replay and automate, applications make a single choice when recording user actions (see also Fig. 4).



- top of basic rate tax; curren
- high earner £100,000 – top

extra_tax
additional tax due to higher rates
= 2500
= income* tax_inc/100
income = 50000
tax_inc = 5

Income	Tax_rate	NI_rate	Loss_tax_allow			ss_
16777.28	20	12	2,500.00			0
30800	20	12	2,500.00			0
50000	20	12	2,500.00	2,500.00	1,141.92	0.00
100000	40	2	5,000.00	5,000.00	141.92	4,997

Figure 3. WS2 explains a value in relation to other data.

3.3 What happened here?

Related to this global history are local *causal* explanations to answer, "*what made* this *happen*". This might be focused on an event in a history or some visible or tangible object. For example, a window or dialog might spontaneously open, but when queried we might find this is due to a cloud file completing its download and opening the app, which itself is due to an earlier double click on the file icon.

Explanations might be focused entirely on a single interface object to answer, "what happened to this object/application?". Track changes in word processors can be thought of as a simple form of this, and in a shared document may include actions by multiple people. Track changes usually says who did what at the level of individual paragraphs or words. Similarly, several early group undo systems used a finer scale of this, allowing selective undo focused on a single graphical object (Berlage 1994).

In the experimental WS2 system, shown in Figure 3, values in the user interface can be individually queried to find out how they were calculated based on other values in the interface (Dix 2024).

3.4 White-box interfaces

WS2 allows authors to easily insert spreadsheet-like capabilities into ordinary web pages or WordPress sites, in a style similar to Victor's reactive documents (Victor 2011) or Apple's OpenDoc.

Often web applications (such as the Google or Microsoft office suites) behave exactly like full screen desktop applications. While this is sometimes appropriate (for example to avoid double scrolling), it is also possible to have web applications that make full use of the web page: scrolling rather than flipping through multiple pages. Automated scrolling after certain actions can still step one through a long web page, but the past inputs are still available simply by scrolling back.

HyperCard, the hypertext/database available on early Mac systems, also had this property as most of the state of an application as well as 'screens' was stored on cards so that it was easy to peer behind the scenes. Indeed this aspect of HyperCard may well have been inspired by Kay's analysis of VisiCalc (Kay 1984), which led him to regard the spreadsheet as an 'ultra-high level language'. Of course spreadsheets have their own problems, but they also in principle have everything available.

There is clear opportunity to learn from Kay's early analysis and cognitive dimension critiques of spreadsheets (Blackwell et al. 2021) to create interfaces that feel more like documents.

3.5 AI is the problem/answer?

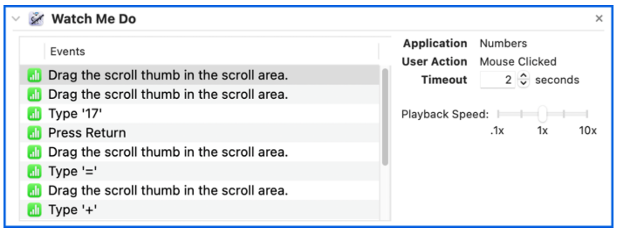
As noted, AI components can lead to problems that drove the need for explainable interfaces as they may act proactively in ways that are hard to predict or anticipate. In some cases this may require the AI components themselves to be explainable; in which case this becomes a standard XAI issue with large bodies of research, but many remaining problems. However within a human–AI system there is an additional higher level of explanation; whether AI is involved in a behaviour, how the AI component came to be invoked in the first place (user action, external event, autonomous agent), what sources of information (personal or external) were taken into account, and whether personal data is exported.

Explaining whether AI is involved, how it was invoked, what is role was and what data were processed where – these are all prominent questions for those dealing with the ethical implications of AI. XUI provides a mode of addressing these. But there are wider ethical implications of AI that XUI has a role in addressing. Human users who feel lost, bewildered or disempowered by technology will be less able to participate actively in discussions about how society deals with the effect on our interactions with each other – whether in the workplace, at home, in our leisure activities or the wider social world. Having an answer to the question, "What just happened?!", also provides a sense of agency and a sense of participation in a technological world.

AI could help in the explanation process, from interpreting user's requests for understanding "why did this happen?", to mining logs of interactions to answer this request.

4. HOW TO DO IT

One of the challenges for user interface architectures is that while the system may be split into functional components, the user sees it all. In the early days of UI monolithic architectures such as Seeheim (Pfaff and ten Hagen 1985) gave way to locally layered components, notably MVC (Krasner and Pope 1988). This is problematic for cross-cutting issues such as the data needed for undo and explanation. Not only may the users interactions cut across multiple components within an application, but also OS and other applications – for example inadvertently clicking on a largely hidden window that then fills the screen. Furthermore, the levels of abstraction may cut across functional components. For example a file selection widget within a word processor that then invokes an OS request for a file, that turns out to be on the cloud.



Drag the scroll thumb in the scroll area.
Drag the scroll thumb in the scroll area.
Type '17'
Press Return
Drag the scroll thumb in the scroll area.
Type '='
Drag the scroll thumb in the scroll area.
Type '+'

Application Numbers
User Action Mouse Clicked
Timeout 2 seconds
Playback Speed: .1x 1x 10x

Figure 4. Apple Automator recording Numbers.

So it may be possible to design some kinds of explanation on an application-by-application basis – particularly important for critical cases such as banking or government sites. However, to deal with some of the problems that less confident users have with phones, such as unexpected app-switching, OS-level support and cross-application standards are needed. Apple's own efforts to encourage applications to factor have had poor traction, even on Apple's own apps (see Fig. 4).

Model-based user interfaces (Meixner et al. 2011) offer another opportunity. Many have not left the lab, but frameworks such as React effectively embody a model of the user interface that could be mined algorithmically or with AI support.

4. OPPORTUNITIES AND CHALLENGES

In a digital world we cannot afford to leave large swathes of the population feeling digitally excluded and blaming themselves for the inadequacies of user interfaces. The fundamental data needed for explainability is similar to that needed for alternative accessible interfaces, AI automation and debugging (Quinn and Alvaro 2025); furthermore, it often already exists for undo support. Indeed, while the Numbers' Automator recording in Figure 4 is at a lexical level, undo says "Edit Cell", and if you invoke undo, it knows which cell and what was edited.

What to do with this data to truly help the user is an even greater challenge. This paper offers some hints, but it is an open area – join the exploration!

https://alandix.com/academic/papers/XUI-2025

ACKNOWLEDGMENTS

This work has been supported by the HORIZON Europe projects TANGO – Grant Agreement n.101120763. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them.

5. REFERENCES

Archer, J., Conway, R. and Schneider, F. (1984). User recovery and reversal in interactive systems. *ACM TOPLAS*, 6: 1-19.

Berlage, T. (1994). A selective undo mechanism for graphical user interfaces based on command objects. ACM TOCHI, 1: 269-294.

Blackwell, A., Britton, C., Cox, A., Green, T., Gurr, C., Kadoda, G., Kutar, M., Loomes, M., Nehaniv, C., Petre, M., Roast, C., Roe, C., Wong, A. and Young, R. (2001). Cognitive dimensions of notations: Design tools for cognitive technology. In *Proc. CT 2001*. Springer, 325–341.

Chilton Computing (2022) *PERQ History.* https://www.chilton-computing.org.uk/acd/sus/perq_history/ (last site build 9/10/2022, accessed 6/6/2025)

Cook, W. (2007). AppleScript. In Proceedings of the third ACM SIGPLAN conference on History of programming languages (HOPL III). Association for Computing Machinery, New York, NY, USA, 1–1–1–21.

Dix, A. (1992). Human issues in the use of pattern recognition techniques, in: *Neural Networks and Pattern Recognition in Human Computer Interaction*, Ellis Horwood, pp. 429–451. https://alandix.com/academic/papers/neuro92/.

Dix, A., Finlay, J., Abowd, G. and Beale, R. (2004). *Human-Computer Interaction*, 3e. Prentice Hall. ISBN 0130461091.

Dix, A. (2022). Follow your nose: history frames the future. Keynote at *AVI 2022: Advanced Visual Interfaces*, Rome, Italy, 6-10 June 2022. https://alandix.com/academic/talks/AVI2022-keynote/

Dix, A. (2024). Just Counting – a tool ecosystem for personal numeric information. *Proc. of AVI 2024*, Article No. 11. doi:10.1145/3656650.3656658

EU (2024). *Council of the European Union. Position of the Council on General Data Protection Regulation*. Technical Report. Council of the European Union, 8 April 2016. http://www.europarl.europa.eu/sed/doc/news/document/CONS_CONS(2016)05418(REV1)_EN.docx/ accessed 01/12/2024.

Fogarty, S. (2005). Binary Opposition. *The Literary Encyclopedia.* (First pub. Feb 2025) ISSN 1747-678x. https://litencyc.com/php/stopics.php?rec=true&UID=122 (accessed 7/6/2025)

Goodman N. (1976). Languages of Art, 2nd edn. Hackett, Indianapolis.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F. and Pedreschi, D. (2018). A survey of methods for explaining black box models, *ACM Computing Surveys* 5:1–42.

Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., Scardapane, S., Spinelli, I., Mahmud, M. and Hussain, A. (2024). Interpreting black-box models: a review on explainable artificial intelligence, Cognitive Computation 16:45–74. URL: https://doi.org/10.1007/s12559-023-10179-8.

Kay, A. (1984). Computer Software. *Scientific American* 251, 3 (sept 1984), 52–59. Also published as Viewpoints Research Institute Technical Report TR-1984-001. https://web.archive.org/web/20171222015552/http://www.vpri.org:80/html/writings.php.

Kluyver, K. Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. Grout, J., Corlay, S., Ivanpv, P., Avila, D., Abdali, S., Willing, C., and Jupyter Development Team. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, Amsterdam.

Krasner, G. and S. T. Pope, S. (1988). A cookbook for using the model–view–controller user interface paradigm in Smalltalk-80. *JOOP*, 1(3).

Meixner G, Paternó F, Vanderdonckt J (2011) Past, present, and future of model-based user interface development. *i-com* 10(3):2–11

Norman, D. (1998). *The Design of Everyday Things*. Cambridge, MA: MIT Press.

Pfaff, G. and ten Hagen, P. eds. (1985). *Seeheim Workshop on User Interface Management Systems*. Springer-Verlag, Berlin.

Quinn, A., and Alvaro, P. (2025). Deterministic Record-and-Replay. *Communications of the ACM*, 68(5), 32-34.

Reason J. (1990). *Human Error*. Cambridge University Press, Cambridge, UK.

Victor, B. (2011). *Explorable Explanations*. http://worrydream.com/ExplorableExplanations/ (dated March 10, 2011, accessed 7/6/2025)