

Patient Interaction

for well-being, productivity and sustainability

Alan Dix

Keynote: FUSION 2024 – Kuala Lumpur, Malaysia, 28th Sept 2024

<https://alandix.com/academic/talks/FUSION2024/>

We live in a world of instant results and fleeting gratification. In HCI no less: the design principles for direct manipulation require immediate feedback and, in the case of graphical actions, sub-second responses. In addition, computers expect us to give them our undivided attention and continually seize it through notifications irrespective of what we are doing or how critical the interruption. This has a clear impact on well-being, and also on productivity as the myth of effective multi-tasking has been comprehensively dissolved. Furthermore, the need for instant and ever more complex computational response has major environmental impacts in terms both of the energy for computation itself and of the digital fast-fashion of discarded devices. Can we reimagine a world of patient human-computer interaction, where interfaces are designed to enable and encourage less feverish use of computational resources and more thoughtful engagement.



patient interaction
for well-being, productivity and sustainability

Alan Dix



Cardiff Metropolitan University



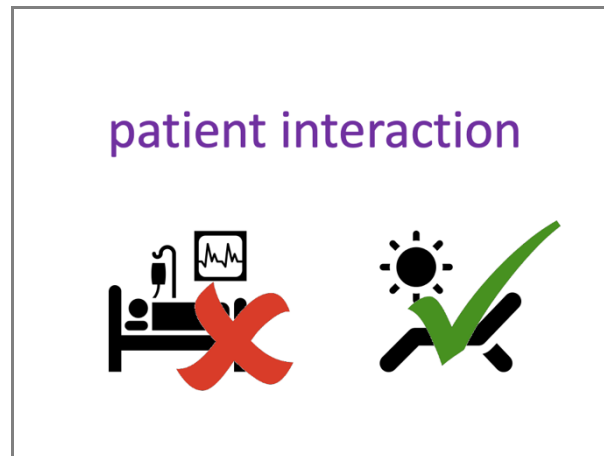
Prifysgol Metropolitan Caerdydd

@alanjohndix



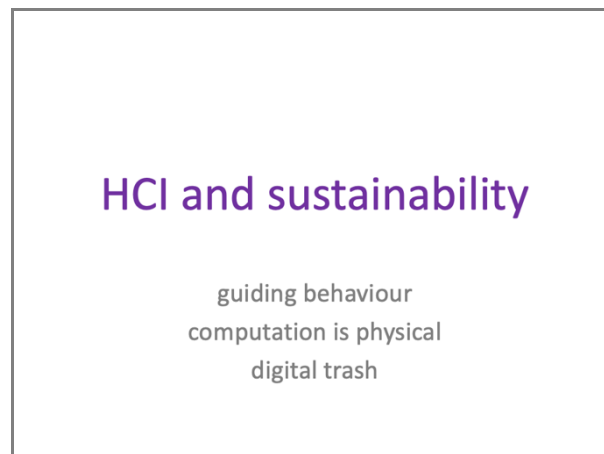
<https://alandix.com/academic/talks/FUSION2024/>

1. Introduction



First a point of clarification, 'patient' in the following is not used in the medical sense, it is about being willing wait for things and more generally being relaxed and thoughtful.

2. HCI and sustainability



The FUSION 2024 theme was “Towards Ethical Technology: Bridging HCI, UI/UX and Sustainability”, and this was the original driver for this talk; however the talk draws in many other strands as well as suggesting a broad new design foci of *patient interaction*.

There is considerable work on sustainability within the HCI literature in general (see [FL24]). We'll take a quick peek at some aspects before seeing how this focus leads to the need for patient interaction.



There are many apps and computer systems that aim to foster better sustainability practices, and this has been a substantial focus within UX design. This includes applications that encourage better energy usage and reuse/recycle apps such as the international Freecycle community and the Olio app. The latter was originally focused on avoiding food waste but now supports giving away or loaning all manner of things.



HCI research also studies more fundamental aspects of psychology or human behaviour that relate to sustainability. I was personally involved in the OnSupply project [FF14,SF15], which focused on awareness of renewable energy availability, prompted by the presence of the island community turbine Tilly. The project included multiple studies on the Isle of Tiree including children’s workshops and installation of devices in a number of island homes. While the goal of the installations was simply awareness a side effect was that of the participants changed their behaviour, doing energy intensive tasks when they knew there was a plentiful supply of renewables.

HCI to help sustainability

reuse/recycle apps

OnSupply (vs. on demand)
goal awareness =>

abundance
plenty when 'in season'
applications in industry and agriculture
... but what about computation




Another side effect of the project was that we began to think about the consequences of a world where there was ever greater reliance on renewable energy. As affordable storage technology has lagged behind production capabilities, there are inevitably times when there is an excess of energy and producers, such as windfarms, are paid to NOT produce energy. While this is usually seen as a problem we thought about it as a potential opportunity. Sustainable energy discussions usually focus on the periods we need to reduce our use. However, we can also think about times of *abundance*, where energy is effectively free [Dx17]. Just as agrarian societies orient themselves around the harvest of different products with feasts and celebration, could modern industry and agriculture re-orient towards periodic uses of energy abundance? We imagined batch-oriented industrial processes and soil-heating cables under polytunnels that are designed to use energy intermittently when plentiful.

Could the same be done for computation?

computation is physical

Bits live in silicon and fibre optics
data centres
high % of global energy use
... even before cybercash
and AI explosion



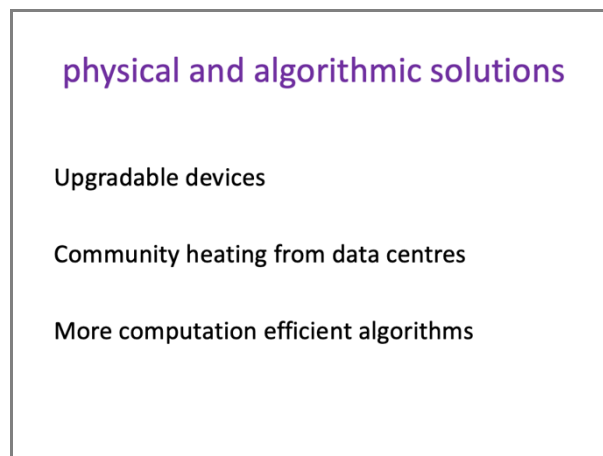
One of the chapters in TouchIT is on “Embodied Computation” [DG22]. Although we often think of computation as abstract, it is always *physical* and embodied, whether in silicon and electrons or graphite on paper.

One consequence of this is the rapidly growing energy use of data centres [HJ23,RJ24,OB24], which was raising concern even before cybercash and the AI explosion



In addition, there is the ever-growing problem of digital trash or e-waste, an estimated 62 million tonnes of it in 2022 [WHO24]. Not only does this use scarce natural resources, the poor disposal of e-waste releases toxic materials, often on the poorest areas of the world.

The relentless throw-away culture of digital devices is fuelled in part by digital fast fashion, but also by the constantly growing demands of greater computational power on tiny devices. The recent AI explosion has exacerbated this with adverts promising a life transformed by AI if you would only buy a new phone! That is the computational demand of software is driving physical obsolescence.



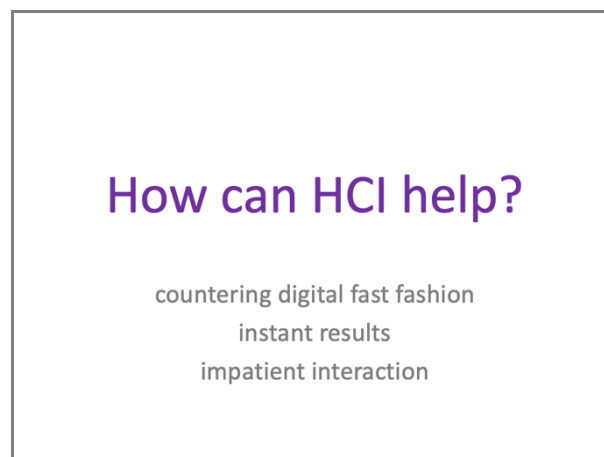
There are physical and algorithmic solutions to ameliorate some of these problems:

- *Upgradable devices* – While desktop computers have long been able to be upgraded with more memory or faster processors, this has been harder in laptops and very rare for phones. To be fair the need to pack greater power into smaller sizes has meant that this was physically impossible at one stage but becoming more achievable now. There are component-based upgradable phones on the market (albeit expensive and a bit clunky!) and the EU legislation requiring devices to be repairable [EP23] may start to change attitudes.
- *Community heating from data centres* – Large data centres may be placed close to local sources of renewable energy, or even under the sea to ease cooling. Some are placed near population centres to provide community heating from waste heat.

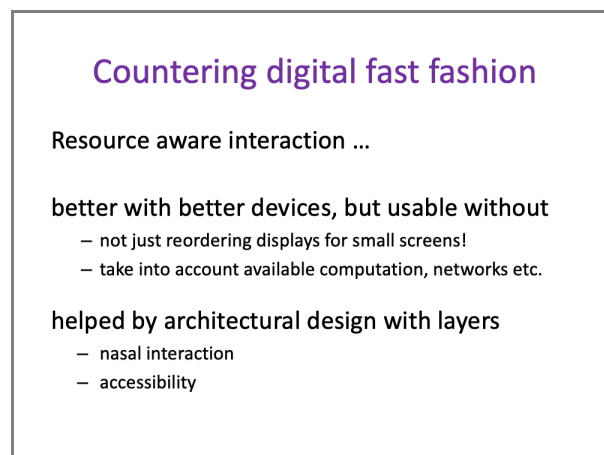
- *More computation efficient algorithms* – There is work on shrinking the footprint of AI and other computation-intensive algorithms driven by the growing awareness of carbon-footprint combined with the need to run on consumer devices. This includes LiGO, which speeds up LLM learning [WP23], LoRA, which creates lower-dimensional internal layers [HS21], memory layers that use fuzzy ‘look-up tables’ [BO24] and most recently Deepseek’s ‘mixture of experts’, which means only small proportion of a network needs to be active at once [LF24].

This is a fast-developing area and in some cases these technology-oriented solutions may reduce the environmental impact without having any effect on the level of service or external behaviour of the system. However, as in any area it helps to take a wider socio-technical view. In other environmental areas the deeper need is to change *demand*, as we saw with the OnSupply project. The same is true for computation, purely technical systems can sometimes achieve the same outcomes with less resources, but the more fundamental issue is changing the desired outcomes.

3. How can HCI help?



So, can HCI help? Are there ways to design interactions that encourage less resource-hungry use, or are more generally environmentally positive.



First, we can help to counter digital fast fashion by designing more resource-aware interactions. When users have better devices, we want to give them better experiences, but still have usable experience when the hardware, software or other resources are not ideal. This will of course not stop consumers wanting the latest shiniest phone but at least mean they do not *have* to do so in order to use your apps.

We are of course used to this for different screen sizes, we expect a responsive web site to work on a smartphone screen, but to use the full screen when on a tablet or laptop. Mobile-first design is not about ‘graceful degradation’ or simply reordering the parts of a web page but having a really full experience on the smallest device and then asking how can this be better [ID25].

The same principle needs to be applied to take into account available computation, networks, or other resources. For example, this may mean performing some computation on the cloud on less powerful devices, as is common in speech interfaces, or using different quality of images or video when network capacity is limited, as seen in the major streaming services. It may also mean designing for gaps in network connectivity, both for areas with no mobile signal, or for areas where fixed and mobile access is fragile. In fact, the first journal paper on mobile HCI, which I wrote in 1995, concerns precisely these issues of intermittent connectivity [Dx95]. Despite growing network infrastructure, these problems persist especially in rural areas [MD14, Dx26]. Some of these issues depend on governmental policy decisions and priorities, and has major social implications [FA13, WG13]. However, this is exacerbated by the network demands of applications; these continually grow, meaning that improvements in connectivity in poorer or marginalised areas are constantly eroded. As designers of interactions, we can change this.

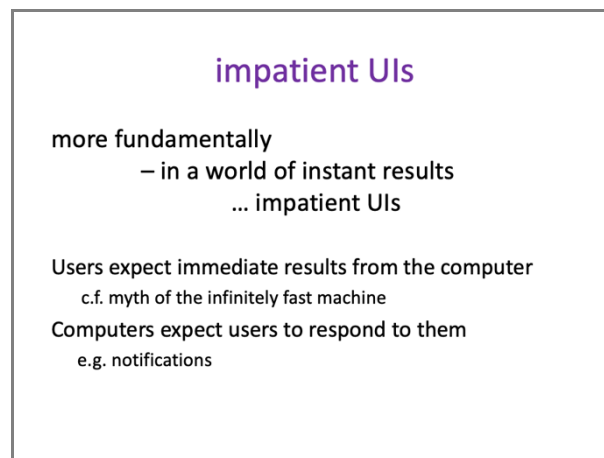
The responsibility does not lie solely with application developers; the main platforms iOS/Apple Store and Android/Google Play Store may put limits on longevity. Studies by Goodwin et al. found that many apps are in fact still usable on older devices even after the manufacturers have ceased support and prevent or discourage downloads [GW23, GW24a, GW24b]. There are issues of security for devices once they cease to be supported, and where this is not a major concern (say a calculator app). However, most users swop well before this point and we can ensure that applications are available for those who choose to continue, thus encouraging longevity and by increasing the user base of older devices mean that OS providers are more likely to maintain security updates.

Some of the solution lies in more careful thinking about what are the most critical parts of each user interaction, what they want to do, what information they need and what actions are most common. In the earliest days of phone-based web interfaces using WAP and nascent smart TVs, I advocated for a mobile or small-screen first approach designing contextual interfaces that offer “the right thing when its needed”, modifying traditional task analysis or interaction flows to identify key paths and user information needs at each step [Dx99]. This accords also with the agile development focus on user stories [Be99] and is related to minimum viable product [Ju00].

This process can also be aided by good architectural design. Implementation frameworks which encourage data layers, such as React [Re25, Nx25], help as these allow alternative front ends to be easily deployed. However, they do not address deeper issues such as different forms of processing, at worst encouraging front-end-heavy interfaces that require powerful end-user devices. Note that architectures that are more flexible for different end-user computational resources are very similar to those needed for varying user capabilities, thus improving accessibility, an issue I’ve addressed in recent keynotes [Dx22, Dx23]

Of course these application-layer frameworks can be enabled by better operating system support, and indeed some issues, notably security, are best addressed here. One could imagine the operating systems designed more akin to the early UCLA Unix security kernel [WK80], so that a small core could be maintained on end-of-life devices allowing applications to

securely access basic services, even when other aspects of the device, for example more sophisticated media services, are no longer actively maintained.

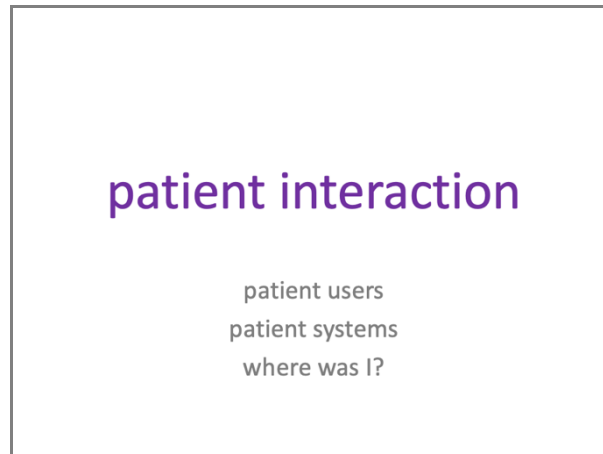


Advocating for resource-aware interaction sounds worthy, but how does it stand up in a world of instant results. We constantly live with *impatient* user interfaces: users expect immediate results from the computer and computers bombard users with notifications the users to also react on demand.

I have written for many years about the former, the way that we live with a ‘myth’ of an infinitely fast machine [Dx87] and the latter, while valuable [SH14], has increasingly become a matter of concern [PR15].



So, what might more patient interactions look like ...



In the rest of this document we'll look in turn at:

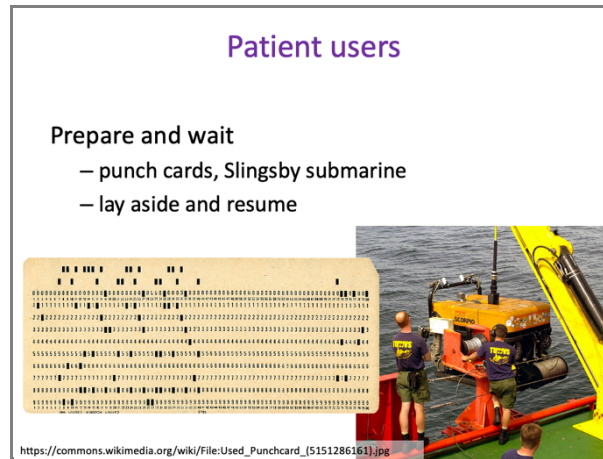
- **patient users** – how interfaces could encourage users to allow more time for systems to work and thus allow better use of computational resources, maybe even so that heavy computation can happen in periods of renewable energy surplus or low demand elsewhere.
- **patient computers** – How we can design interactions that allow users to interact when it is convenient for them.

Both often break the lockstep turn-taking dialogue that is most familiar in user interfaces. This can create issues of situational awareness “**where was I?**” moments, so we also consider ways to help re-engage when using slow interfaces.

4. Patient users

The desire for instant results inevitably leads to either more computational power locally, with attendant device redundancy and e-waste, or remote computation, such as LLMs, that is energy intensive, but not necessarily when that can be provided by renewable sources. However, note that in resisting this cult of the instant, the intention is not to berate users for being impatient, but thinking about ways in which interaction design does not force them to be so.

Most of the examples in this section concern programming or data science, but with the rise of LLMs, we are all using large-data and large-compute resources, so hopefully we can learn lessons for all types of interaction.



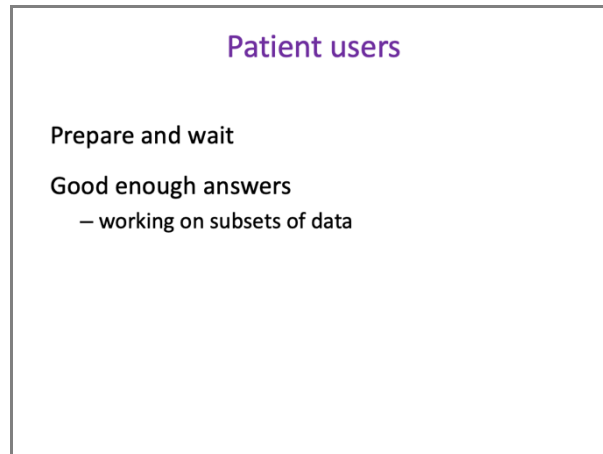
Results were not always instant. In the early days of computer programming code was written onto punch cards and fed into the computer in batches. Often magnetic tapes had to be retrieved and loaded in order to run a program. If you visited programmers at that stage, you would find no terminals or screens, just desks with people hunched over paper coding sheets and using various aids such as flowcharts to help them plan the overall flow of execution alongside fine-grained practice such as documenting expected values after each line of code.

In the 1990s, well into the era of the desktop PC, I worked as a consultant with an engineering company Slingsby who made small submarines for oil and telecoms industry, mostly remote controlled via tethering cables. The submarine used an embedded processor and attached Texas Instruments DSP. Early development used a plug-in board on a PC, which enabled a fairly standard development cycle, but once deployed in the pressurised container on the submarine, everything to and from went through a 30 cps (yes thirty characters per second) link in the tethering cable.

Days before the first sea trials a critical bug emerged leading to a 36-hour three-person non-stop debugging session. Downloading code through the 30cps link took more than 20 minutes and with the rather slow compiler for the specialised chips, a complete debug cycle was around 40 minutes. Added to this, the logging information back through the link was limited to single characters produced at key points in the code. Boy did we think hard about each code–compile–download cycle and continue to think during the long waits between cycles.

As a coder I have no desire to go back to the days of punch cards or 40-minute debug cycles, but it is too easy to get caught up in the rapid cycles so that one forgets to sit back and think. Maybe there are lessons to learn from this *prepare-and-wait* style of working, with highly interactive tools to help preparation, but accepting gaps between.

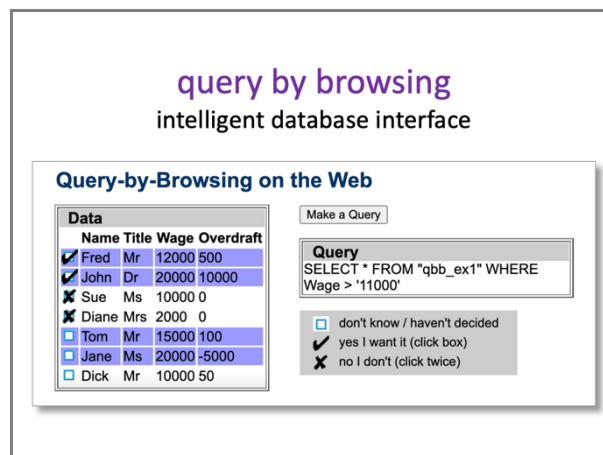
Indeed, these issues have not gone away. If you are working on the Mars Rover or a deep-space mission, similar issues arise. The comms links have higher bandwidth (albeit far less than terrestrial links) but the time delay between Earth and Mars is between 5 and 20 minutes depending on their relative positions around the sun, so ‘suck it and see’ programming doesn’t cut it! Similar issues arise for big AI models, which can take days to run.



Data science is big business and of course uses big data. In computing we often feel we need the *optimal* solution using *all* the available data. However a key lesson from studying human and animal cognition is satisficing [Si56], the way in which we often find good enough solutions.

Many traditional algorithms and AI have to work this way because the problems are intrinsically NP-hard or of a size that optimal calculations are impractical. However, it can be applied as a general design heuristic, both for purely automated algorithms [AK24,GS00] and human-AI hybrid systems [DR24,TG25].

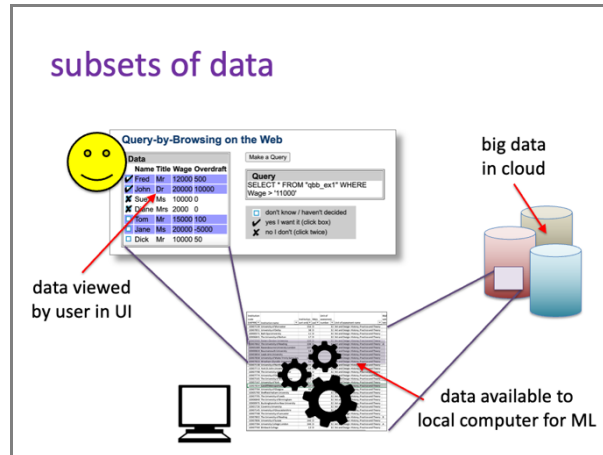
One variant of this for big data is to work on subsets of big data and only move to larger portions when needed. As an example of this, let's look at Query-by-Browsing.



Query-by-Browsing (QbB) is an intelligent database interface that generates queries for you based on user-selected records. It was originally developed as a thought experiment in my (very) early work on the potential for social, ethnic and gender bias in black-box machine learning systems [Dx92a], but when implement turned out to be useful [DP94], and now has a web version you can try for yourself (<https://alandix.com/labs/qbb/>).

The basic idea is simple. The user selects records of interest in the right-hand pane that lists a database table (tick for wanted and cross for not wanted). When the user asks, the system uses a variant of ID3 (or other algorithm) to create an SQL query (right-hand pane) that matches the selected records. The records matching the query are then highlighted on the left-hand pane and the use can select more records if the query is not as wanted.

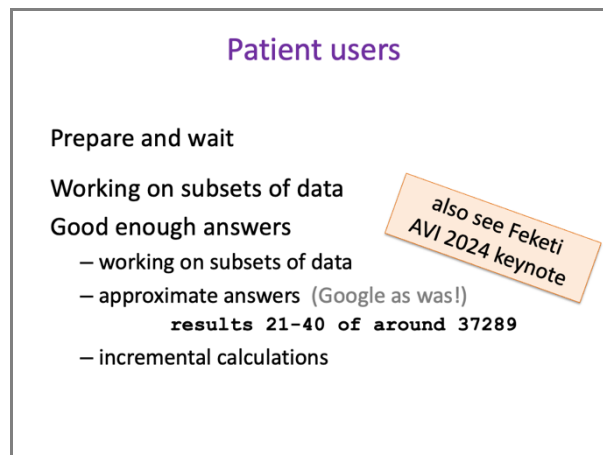
The system hinges in the fact that a user may be able to recognise that the query is correct even if they would struggle to write it from scratch themselves. The dual use of intentional (SQL) and extensional (highlighted records) representations is also crucial as different aspects of the generated query are easier to verify in one representation or the other.



All QbB implementations to date have all the data available. In the web version (ID3 variant) only the user-labelled records are used in the ML algorithm, but a semi-supervised algorithm might use the user-labelled data together alongside the larger set of unlabelled records. Also, one item on the QbB roadmap is for the AI identify uncertain parts of the decision rules and use these to suggest records that the user might want to check. This would require more data than the user-selected records.

If applied to big data one could imagine different subsets of the data being used for different aspects of the user interaction. The smallest subset is the records in the user interface listing. It is these records that the user scrolls through and labels. The machine learning algorithm might have a far larger set of data to be used for semi-supervised learning or for choosing the problematic records to show to the user. Finally, this is itself a subset of the full dataset stored in the cloud.

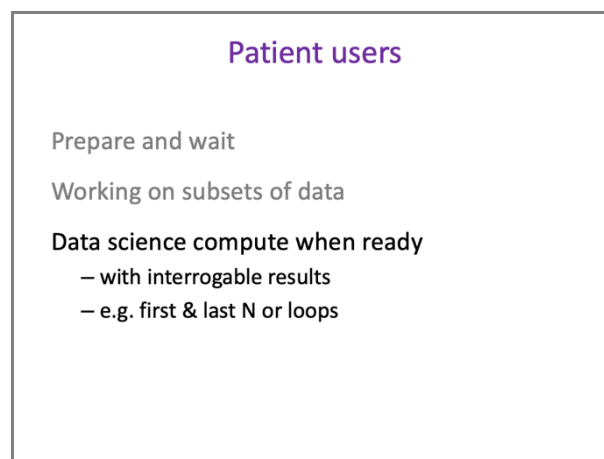
Eventually the whole dataset may need to be queried or updated based on the generated query, but the user's interactions are faster and less computationally hungry than if the whole dataset were used continuously.



It is often sufficient to provide approximate answers, and indeed it is often better to have an approximate answer quickly than an exact one later. Until recently Google search results used to have a message that said, “results 21-40 of around 37289”. The ‘around’ figure was an estimate and could change as you paged through the search results. Google could have found every result at the beginning and then given an exact answer for the number of hits, but this would have meant finding maybe many thousands of results that you would never page through far enough to see. The approximate answer is good enough for the user’s interactions, whereas finding every result upfront would have been wasteful of computer resources and resulted in a painfully slow UI.

Incremental calculations are widely used in engineering domains, that is where the algorithm returns an initial answer after a short time and then provides progressively better answers if you wait longer. This is a natural result of iterative algorithms, such as gradient descent, that continually improve estimates, or stochastic algorithms, such as hill climbing with random starts, which run the same calculation many times looking for the average or best result.

Jean-Daniel Fekete and colleagues have been using techniques rather like those suggested for QbB above but applied to pure machine learning algorithms and visualisation [FF24,RP24], and Geoff Ellis and myself adopted sampling to enable faster and more responsive visualisation [DE02, ED02]

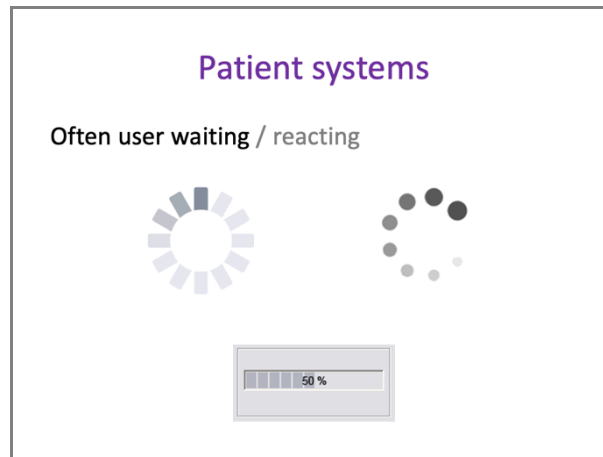


Some algorithms, notably those operating on graphs, are hard to trial on samples of data. Imagine a graph of ‘closest friends’, where each person has only 5–10 connections. If we were to do a 1 in 1000 sample of the complete population, the vast majority of people would have no connections at all in the sample. To deal with this Geoff Ellis and I imagined ‘Coronation Street’ samples (named after a soap opera), a kind of snowball technique, that given a sample point would also bring in neighbours, but this would still struggle, for example, with path-like connections of the ‘six degrees of separation’ kind.

If given a large data or compute question, we simply say to the system “compute this when you are ready”, then it is hard to later interrogate the results, in particular, the kind of step-by-step logging and debugging one gets from more interactive execution. There has been a long-standing literature on the use of checkpoints to allow replay and roll-back of code including the libckpt Unix library on the 1990s [PB95, WH95]. These have been especially important for applications that are event-based [MV16], multi-threaded [PT03] or distributed. It is especially important for distributed applications [MT25]. Furthermore recent techniques can achieve this with low overhead [BP18]. Similar techniques could be applied to these large “run when ready”

computations, for example, storing data for the first and last few iterations of vast loops, with sampled points between.

5. Patient systems

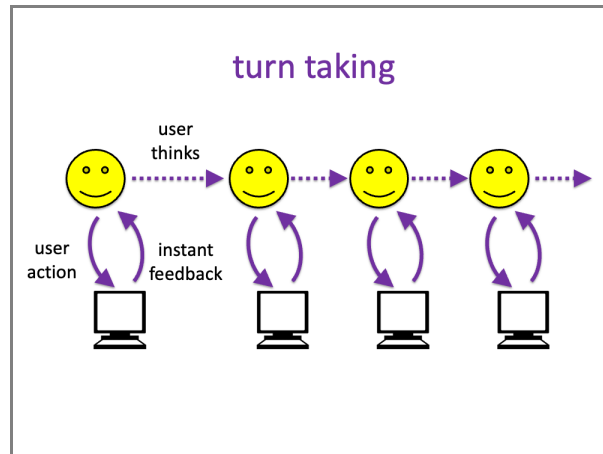


The opposite side of the coin is that computer systems may leave you waiting while they perform some long-lasting task.

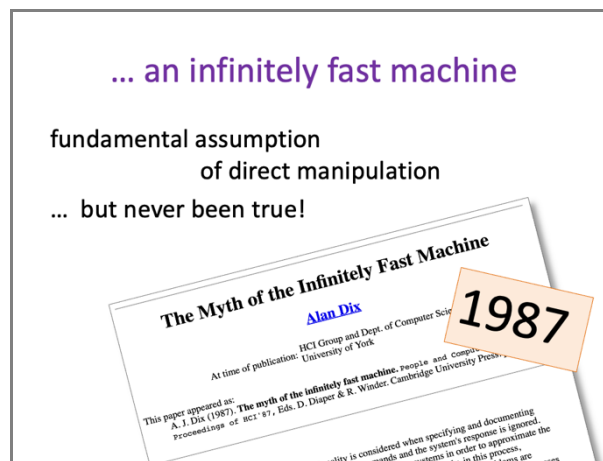
This is bad both in terms of computer usage and user experience. For the former, because the computer designer knows you will be waiting and so is trying to do the computation as fast as possible – an extension of the impatient user problem. For the latter ... well, we have all sat there watching a little “I’m busy” spinner or progress bar grind its way painfully slowly towards completion!

We’ll focus on the user experience, but paradoxically if we do this then the urgency for the computation reduces hence addressing the sustainability issues also.

Some fixes just make the waiting less bad. There is extensive research on different kinds of progress indicators this includes detailed manipulations such as non-linear rates of movement [HA07] and different styles of patterning [HY10] and shape [OY14]. More radical research has looked at methods to help users use the idle time [HC12] or playing well-known songs to help users estimate time to completion [KP06].

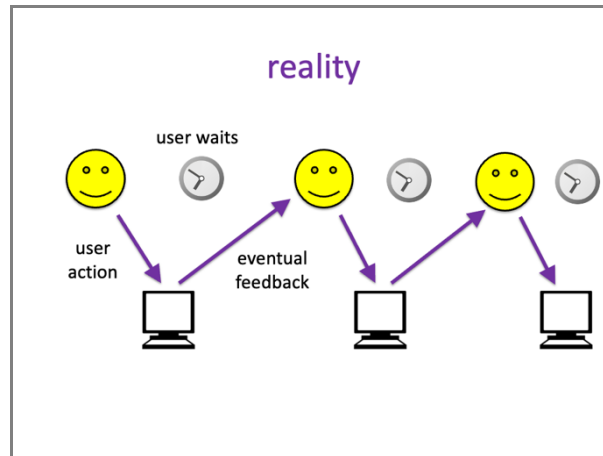


This is important, but points to a much more fundamental problem, that is the notion of turn-taking, which is taken for granted within so much interaction design. It is at the heart of Norman’s highly influential execution–evaluation loop [No90] and a tacit assumption behind design and evaluation principles such as Shneiderman’s golden rules [Sh87] or Nielsen’s heuristics [Ni94].



Crucially the notion of direct manipulation, which has been central to the imaginary of HCI since its earliest days, is built on the idea that objects in graphical user interfaces can behave almost as if they were physical with instant reactions to user actions [Sh82,HH86].

However, this has never been true – in reality all computation takes time. Sometimes this can be fast enough to be effectively instantaneous, but often we need to design taking small or large delays into account. This is an issue I first described in 1987 as the “Myth of the infinitely fast machine” [Dx87], and which I’ve returned to repeatedly over the years [Dx20a].



Rather than action-feedback cycle of direct manipulation, real turn-taking interaction should be seen more as a series of cycles of the form:

1. user performs action
2. user waits – as system updates things, performs cloud access, etc.
3. eventual feedback

... an infinitely fast machine

fundamental assumption
of direct manipulation

... but never been true!

max delay before feedback:

hand-eye coord – 200ms e.g. trackpad

major action ~ 1-2 secs

... breakdown

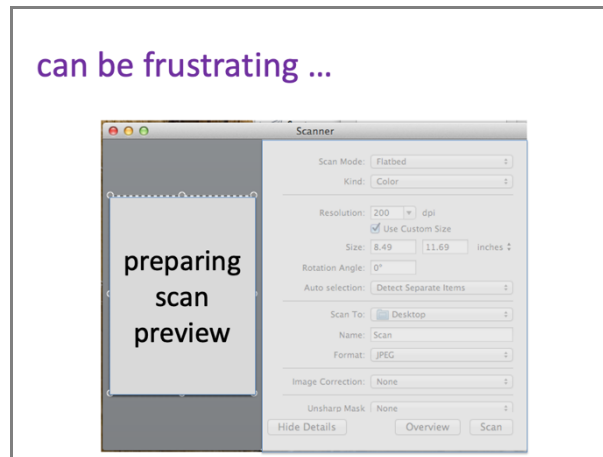
The length of the delay is crucial in terms of the way it interacts with the human perceptual, motor and cognitive systems.

Our hand-eye coordination system is built to deal with the delays in the internal perceptual and motor processing paths within the human body, but delays of more than a couple of hundred milliseconds lead to rapid breakdown of any form of mouse or finger tracking such as dragging a window or resizing a photo [Sh84,Dx26]. This effectively means that feedback for this form of response has to be local.

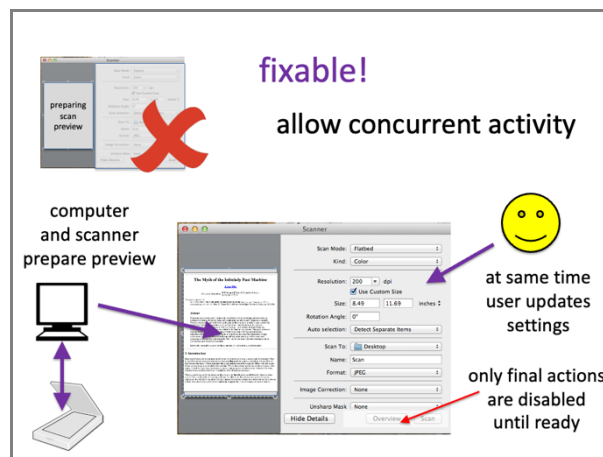
For actions such as the response to a menu selection, the feedback can be a little longer. Our internal idea of ‘now’ is about 1 second long, so that responses faster than this are felt as ‘instantaneous’. Things that take longer than a second or so are no longer connected unconsciously. to begin to feel more like act-wait-feedback.

‘Large’ actions such as moving to a new web page, can take a bit longer. An early review Shneiderman found that waits of 5-10 seconds could be accepted, but rapidly gave way to a sense that something was broken and shorter waits of a only a few seconds better. While the

technology has changed, SEO recommendations still suggest 2.5 second maximum for web page rendering ... however, we have all experienced both web and desktop apps that are slower!



Sometime this can just be frustrating. When you open the scanner dialog in MacOS, the system initiates a low-resolution overview scan in order to help you choose the scan area. This can take 15-30 seconds depending on the scanner and document. During this time the rest of the dialog is disabled. Often there are things you could do whilst waiting for the preview, or when you know the document fills the scanner area, you might not even care about the preview. However, you cannot set the target filename, change the scan resolution settings, or swop between flatbed and document feeder options until the computer decides it is ready. You are forced to wait for the computer and react when it wants.

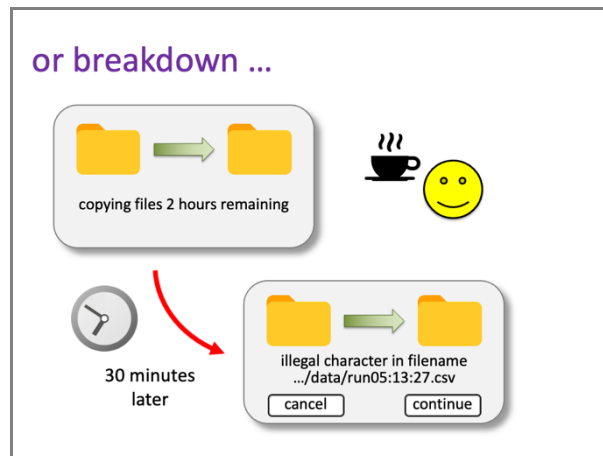


This is very easy to fix by allowing the user to concurrently access the settings in the dialog while the system is performing the preview scan. If some options cannot be performed during the scanning, those options, and those options only, should be disabled.

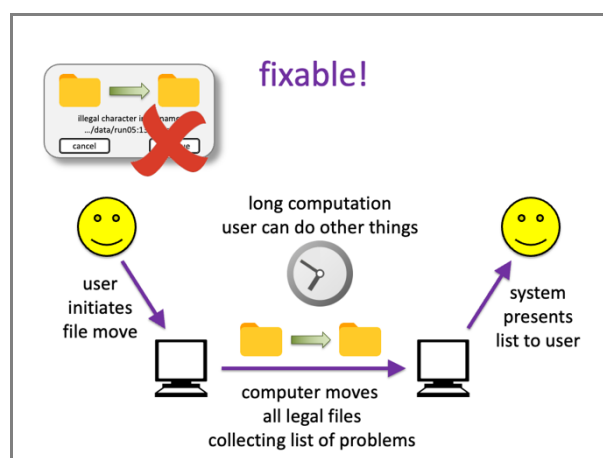
As a general design heuristic, one should, as far as is reasonable, allow the user to interact concurrently while the system works unless the delay is guaranteed to take no more than a few seconds.

The MacOS scanner dialog is by no means an isolated example, when I was walking around Wales [Dx20b], I wore a medical-grade ECG device. Data from this needed to be uploaded via a special device that then communicated with bespoke software on a Windows laptop. The

upload was over a serial line and took 10-15 minutes. When ready it popped up a message asking you to decide what to do next: whether to process further inside the upload software (for example, to visualise the heart rate over time), or simply to save to disk. However, if you didn't interact with it fast enough, after about five minutes the message would time out and the upload was cancelled – you had to start the whole process again. Because of this, you had to constantly monitor the upload, unable to concentrate on anything else.



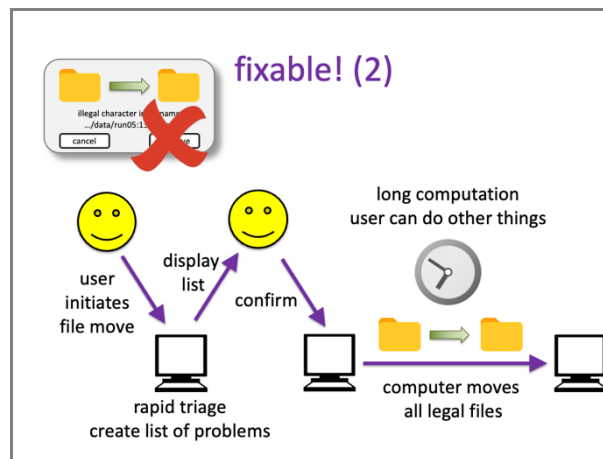
Another example, where there can be a complete breakdown is when you copy a large number of files between devices, for example, when performing a backup to a removable hard drive. You initiate the copy and after a while the system will offer an estimate as to how long the copy operation will take. This is of course very helpful, and if, for example, it says the copy will take two hours, you know you can go away, maybe make yourself a cup of tea and do something else. Half an hour later you take a peek to see how the copy is proceeding, expecting it to say 25% complete with another hour-and-a-half to go, but instead, maybe just minutes after you left, the system had encountered a filename that for some reason the backup disk format can't accept and the copy has stopped asking you what to do next. Instead of being able to get on with something else, you have to constantly keep an eye on the computer to press buttons when it gets stuck. Once again, the computer expects you to be there waiting for it when it is ready for you to interact with it.



As with the MacOS dialog, this problem is fixable.

Some years ago, in the late 1990s, I supervised a student project at Staffordshire University, where the student first implemented the copy dialog in the most obvious way, a loop that copied at each file in turn and stopped to ask the user every time it encountered an issue.

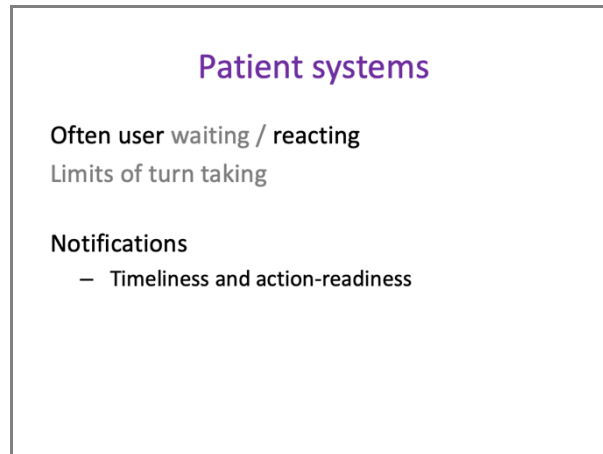
He then modified the code so that the system instead built up a list of problem cases and only when finished popped up a dialogue saying, “what do you want to do with these?”. This meant that the user could safely go away and ignore the system while it performed the long copy process, knowing that it would run to completion without needing any interaction until the end.



As a second experiment, he modified the code so that it first rapidly scanned the disk checking the filenames. This only took a few seconds. The system was then able to ask the user what they wanted to do (ignore the problematic files, rename the, etc.). Once this was confirmed, the system was able to run through the lengthy copy process, again allowing the user to get on with other things.

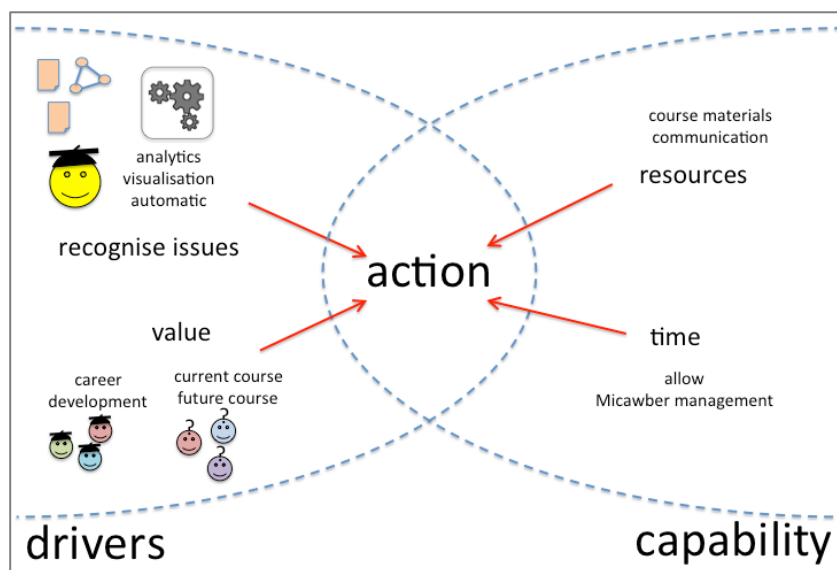
Crucially neither modification was substantially more complicated to code than the original implementation that demanded the user’s constant availability.

Incidentally, I have found the second approach useful algorithmically even when there is no user interaction required. Often loops that update data structures can get very complicated with multiple internal decisions, especially if the data that is being iterated over is also updated within the loop. Nowadays I often use a two-pass to-do-list solution, with one pass iterating over data and building a list of what needs to be done, followed by a second pass through the list doing the actions. As well as often being clearer to understand, it is also often a more robust approach as many potential problems are discovered during the first pass, meaning it is possible to exit with no change rather than with a partially updated data structure.



As well as keeping us waiting doing nothing, impatient systems often try to force us to react to them when they want ... notifications!

Although the process had already started with email, the ubiquity of notifications means that our lives are based almost entirely around responding to computer demands, rather than following through considered plans of action. Good interaction requires a match between the pace of the task and the pace of communication [Dx92b], but most notification systems are largely one size fits all demanding action *now*. Furthermore, most do not make taking that action easy.



Drivers and capabilities for analytics-driven academic action (from [DL15])

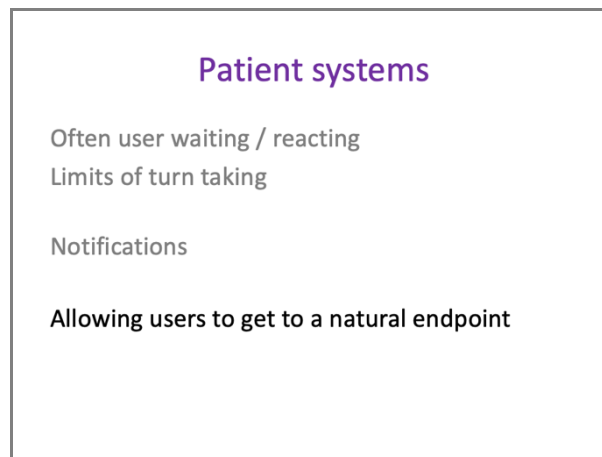
This is an issue in many areas, not least education. The Holy Grail for learning analytics systems is ones that can deliver actionable insights, such as indications of students in trouble at a point when remedial actions is possible [TA11, SP16]. Many of these systems are dashboard based, which you consult when (if) you remember.

While working at Talis (now part of Sage), we were interested in the potential to deliver more notification-based prompts. However, it would not be appropriate to send a ping to your phone, when you are halfway to a lecture to say it has just detected that a student's grades suggest

action is needed. This insight probably requires an email prompt the next morning, whereas insights into potential improvements of your course for next year need to come when you are in the process of reviewing the material, not when it is in the middle of exam marking season.

In general, action requires some sort of *trigger* – the purpose of the notification – but also that, when it occurs, the recipient has the right *drivers or motivation*, the necessary *information* resources and crucially *time*.

This is rather like the design of mobile-first applications just at a larger time scale – as designers we need to think about the user’s long-term activities, how the things that require notifications fit into that, what might be an appropriate pace for notifications, and making sure the right information is available.



Notifications have two problems

- they demand you do what the computer wants instantly
- they interrupt what you are already doing

Interruptions have been studied for many years in the HCI literature, and their impact is rarely positive [RH94, MG08]. The most intrusive notifications, especially auditory alarms or modal dialogs, can interrupt your train of thought and at worst mean that you forget that carefully crafted phrase you were about to write. In general, interruptions involve setting down one task to begin another. This putting down of a task is often difficult, both intrinsically, and because usability and design guidelines are focused almost entirely on the periods of engagement with an application, not the gaps between. In my own work I have looked at extending heuristics to include the way that one needs to stop and restart interactions [Dx20a] but, even then, assuming the user has control.

When designing the timing of notifications, we should give thought to how these allow users to come to *natural endpoints* in interaction.

6. Where was I?

Where was I?

patient user

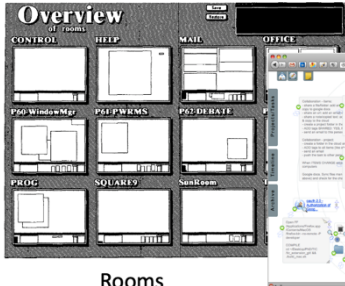
- lay aside and resume for prepare and wait

patient system

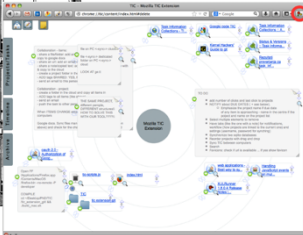
- action-readiness for timely notifications

Thinking about patient users led us to consider prepare-and-wait interaction paradigms. Of course, after this wait the user may have lost track of the original task. Similarly, when thinking about patient systems and notifications we have considered the need for techniques to make it easy for a user to lay down the thing they are doing at a natural point and also have the right resources to act on a notification when it is appropriately delivered.

having the info you need to re-start



Rooms
(Henderson & Card 1987)



TIC
(Kljun, 2013)

All of these need “where was I?” information to help users re-start activities where they left off.

In the very early days of graphical user interfaces, Henderson and Card recognised the cognitive load of human task switching and so used an analogy with computer working sets of memory to create the Rooms interface [HC86]. This collected all the open windows and applications for a task together allowing the user to return to a screen in the same configuration as when they last were last doing it. More recently Matjaz Kljun created a browser extension TIC (Task Information Collection), which allowed users to gather information resources related to tasks [Kl13, Kl25].

At a finer level effective layering of system architecture, as described earlier, could help the saving and restarting of interaction state, which is often lost even when files and documents are saved.

Where was I?

patient interactions

- lay aside and resume for prepare and wait
- action-readiness for timely notifications

having the info you need to re-start

N.B. also helps impatient interruptions!

Of course, having better facilities of this sort will also help impatient interruptions and multi-tasking ... which, in reality, always means rapidly swapping single tasking and is never efficient!

7. Summary

summary

designing user interaction
to counter digital fast-fashion

in a world of instantaneity design for

- patient users
 - letting computation happen when resources allow
- patient systems
 - letting users do things at their pace

In summary we have discussed how appropriate user experience design can start to counter digital fast fashion. This is partly about doing more with less computational resources and adopting a design strategy that is least-first, that is first creating effective designs on minimal hardware and only after that thinking about yet better experiences if more power is available. As part of this we saw how in a world centred around instantaneity our design often also leads to impatient interfaces, which are intrinsically resource heavy and poor for well-being.

Good design can encourage *patient users*, which do not demand that systems respond instantly to everything. Instead, prepare-and-wait interaction styles intersperse periods of rich interaction with periods when the system works. This is especially important in data- or resource-hungry applications, but with AI that increasingly means everything.

Similarly good design can create *patient systems*, which do not demand that users wait around while the system works and which do not assume the user will respond instantly to system notifications. Instead, systems can use list-building interaction styles, that either rapidly triage large data to find potential interaction points up front or store lists of queries for when the user

is ready. Notifications likewise can be designed to match their pace and urgency so that users can deal with them when their current task reaches a natural end.

In both cases there is a need for users to put down and take up tasks after periods doing other things. Systems need to be designed at both an architectural and user interaction level so that this is possible and easy.

These are just first thoughts in potentially a large area with wide potential for research and practical designs for patient interaction. The benefits are multifaceted including human well-being, commercial productivity and environmental sustainability.



Acknowledgements

Parts of this work have been supported by the HORIZON Europe projects TANGO - Grant Agreement n. 101120763 and SoBigData++ Grant Agreement n. 871042. 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.

Many thanks also to the organisers of Fusion 2024 for inviting me to give this keynote and the attendees for their insightful post-talk questions.

References

- [AK24] Arumugam, D., Kumar, S., Gummadi, R., & Van Roy, B. (2024). Satisficing exploration for deep reinforcement learning. arXiv preprint arXiv:2407.12185.
- [Be99] Kent Beck (1999). Extreme Programming Explained: Embrace Change. Addison-Wesley. ISBN 9780201616415. OCLC 41834882.
- [BP18] Jonathan Bell and Luís Pina. CROCHET: Checkpoint and Rollback via Lightweight Heap Traversal on Stock JVMs. In 32nd European Conference on Object-Oriented Programming (ECOOP 2018). Leibniz International Proceedings in Informatics (LIPIcs), Volume 109, pp. 17:1-17:31, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2018) DOI:10.4230/LIPIcs.ECOOP.2018.17
- [BO24] Berges, V. P., Oğuz, B., Haziza, D., Yih, W. T., Zettlemoyer, L., & Gosh, G. (2024). Memory Layers at Scale. arXiv preprint arXiv:2412.09764.

- [Dx87] A. J. Dix (1987). The myth of the infinitely fast machine. People and Computers III - Proceedings of HCI'87, Eds. D. Diaper & R. Winder. Cambridge University Press. pp. 215-228.
- [Dx92a] A. Dix (1992). Human issues in the use of pattern recognition techniques. In Neural Networks and Pattern Recognition in Human Computer Interaction Eds. R. Beale and J. Finlay. Ellis Horwood. 429-451. <https://alandix.com/academic/papers/neuro92/>
- [Dx92b] A. J. Dix (1992). Pace and interaction. Proceedings of HCI'92: People and Computers VII, Eds. A. Monk, D. Diaper and M. Harrison. Cambridge University Press. 193-207. <https://alandix.com/academic/papers/pace/>
- [DP94] A. Dix and A. Patrick, 1994. Query By Browsing. Proceedings of IDS'94: The 2nd International Workshop on User Interfaces to Databases, Ed. P. Sawyer. Lancaster, UK, Springer Verlag. 236-248.)
- [Dx95] A. J. Dix (1995). Cooperation without (reliable) Communication: Interfaces for Mobile Applications. Distributed Systems Engineering, 2(3): pp. 171-181. <https://www.alandix.com/academic/papers/DSE95/>
- [Dx99] A. Dix (1999). Hazy, Crazy, Lazy Days are Over - time for designers to think. keynote Designing Information for Mobile and Broadband Network Users, London, 15th December 1999. <https://alandix.com/academic/papers/mobile-and-b-99/>
- [DE02] A. Dix and G. Ellis (2002). By chance - enhancing interaction with large data sets through statistical sampling. Proceedings of Advanced Visual Interfaces - AVI2002, Trento, Italy, ACM Press. pp.167-176. <https://www.alandix.com/academic/papers/avi2002/>
- [DL15] Dix, A., & Leavesley, J. (2015). Learning analytics for the academic. Journal of Universal Computer Science, 21(1), 48-65.
- [Dx17] Alan Dix (2017). Abundance: a new approach to energy sustainability. Tiree Tech Wave Working Paper. <https://tireetechwave.org/projects/abundance/>
- [Dx20a] Alan Dix (2020). Taking the Long View: Structured Expert Evaluation for Extended Interaction. Proceedings of AVI 2020 (Advanced Visual Interfaces), ACM. DOI: 10.1145/3399715.3399831 <http://alandix.com/academic/papers/AVI2020-long-view/>
- [Dx20b] A. Dix (2020). The Walk Exploring the Technical and Social Margins. HCI Outdoors: Theory, Design, Methods and Applications. McCrickard, Scott, Jones, Michael, Stelter, Timothy (Eds.) pp. 19-50
- [DG22] Alan Dix, Steve Gill, Devina Ramduny-Ellis, and Jo Hare (2022). Embodied Computation. Chapter 17 in TouchIT: Understanding Design in a Physical-Digital World. Oxford University Press, 2022. <https://physicality.org/TouchIT/>
- [Dx22] A. Dix. (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/>
- [Dx23] A. Dix. (2023). Exceptional Experiences for Everyone. Invited talk at AMID 2023 – 1st International Workshop on Accessibility and Multimodal Interaction Design Approaches in Museums for People with Impairments, 26 Sept 2023. CEUR Vol 3622, pp.1-6. <https://alandix.com/academic/talks/AMID2023-exceptional/>
- [DR24] Alan Dix, Matt Roach, Tommaso Turchi, Alessio Malizia and Ben Wilson (Editors) (2024). Designing and Building Hybrid Human-AI Systems – SYNERGY 2024 Proceedings of the 1st International Workshop on Designing and Building Hybrid Human-AI Systems. co-located with 17th International Conference on Advanced Visual Interfaces (AVI 2024) Arenzano (Genoa), Italy, June 3rd, 2024.. CEUR Workshop Proceedings, Vol. 3701. <https://ceur-ws.org/Vol-3701/>
- [Dx26] A. Dix. (2026). Network-Based Interaction. To appear as Chapter 12 in J. Jacko (Ed.), The Human-Computer Interaction Handbook (4th Edition). CRC Press, 2026. <https://alandix.com/academic/papers/network2026/>
- [ED02] G. Ellis and A. Dix (2002). Density control through random sampling : an architectural perspective. Proceedings of the Sixth International Conference on Information Visualisation - IV02, London, UK, July 2002, IEEE Computer Society. pp. 82-90. <https://www.alandix.com/academic/papers/IV2002/>
- [EP23] European Parliament (2023). New EU rules encouraging consumers to repair devices over replacing them. European Parliament News. Press Releases 21 Nov. 2023. <https://www.europarl.europa.eu/news/en/press-room/20231117IPR12211/>
- [FF24] Jean-Daniel Fekete, Danyel Fisher, Michael Sedlmair (2024). Progressive Data Analysis. Eurographics, pp.231. DOI:10.2312/pda.20242707

- [FF14] Maria Angela Ferrario, Stephen Forshaw, Peter Newman, William Simm, Adrian Friday, Alan Dix (2014). On the edge of supply: designing renewable energy supply into everyday life. The 2nd International Conference on ICT for Sustainability (ICT4S 2014), Stockholm, Sweden 24-27 August 2014. DOI:10.2991/ict4s-14.2014.6
- [FA13] M. Fourman, A. Alexander, F. Bechhofer, J. Brown, N. Macaskill, J. Moore, N. Osborne, S. Skerrat and M. Wade (2013). Spreading the benefits of digital participation: An interim report for consultation. Edinburgh, Royal Society of Edinburgh: 22-24. <https://www.research.ed.ac.uk/en/publications/digital-scotland-spreading-the-benefits-of-digital-participation->
- [FL24] Adrian Friday, Ann Light, Jason Tarl Jacques, Matthew Louis Mauriello, Kathrin Gerling, Robert Soden, Gözel Shakeri, Nicola J. Bidwell, Vishal Sharma, Neha Kumar (2024). There is no plan(et) B: On sustainability and HCI. Interactions Blog, December 23, 2024. <https://interactions.acm.org/blog/view/there-is-no-planet-b-on-sustainability-and-hci>
- [GS00] Goodrich, M.A., Stirling, W.C. & Boer, E.R. Satisficing Revisited. *Minds and Machines* 10, 79–109 (2000). <https://doi.org/10.1023/A:1008325423033>
- [GW23] Goodwin, C., Woolley, S., de Quincey, E., Collins, T. (2023). Quantifying Device Usefulness - How Useful is an Obsolete Device?. In: Abdelnour Nocera, J., Kristín Lárusdóttir, M., Petrie, H., Piccinno, A., Winckler, M. (eds) *Human-Computer Interaction – INTERACT 2023*. INTERACT 2023. Lecture Notes in Computer Science, vol 14145. Springer, Cham. https://doi.org/10.1007/978-3-031-42293-5_8
- [GW24a] Goodwin, C. and Woolley, S. (2024) 'Barriers to device longevity and reuse: A vintage device empirical study', *Journal of Systems and Software*, Elsevier, 211, p. 111991.
- [GW24b] Goodwin, C. and Woolley, S. (2024). "Should I Throw Away My Old iPad?" - Reconsidering Usefulness in Obsolete Devices. In: Bramwell-Dicks, A., Evans, A., Winckler, M., Petrie, H., Abdelnour-Nocera, J. (eds) *Design for Equality and Justice*. INTERACT 2023. Lecture Notes in Computer Science, vol 14535. Springer, Cham. https://doi.org/10.1007/978-3-031-61688-4_32
- [HA07] Chris Harrison, Brian Amento, Stacey Kuznetsov, and Robert Bell. 2007. Rethinking the progress bar. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)*. Association for Computing Machinery, New York, NY, USA, 115–118. <https://doi.org/10.1145/1294211.1294231>
- [HY10] Chris Harrison, Zhiquan Yeo, and Scott E. Hudson. 2010. Faster progress bars: manipulating perceived duration with visual augmentations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. Association for Computing Machinery, New York, NY, USA, 1545–1548. <https://doi.org/10.1145/1753326.1753556>
- [HJ23] Ian Hodgkinson, Lisa Jackson and Tom Jackson (2023). On track for 6.8 billion years of continuous movie streaming: Data, energy & need for digital decarbonization. OECD Observatory of Public Sector Innovation (OPSI), 6 April 2023. <https://oecd-opsi.org/blog/digital-decarbonization/>
- [HC12] Christophe Hurter, Benjamin Cowan, Audrey Girouard, Nathalie Riche. Active progress bars : aiding the switch to temporary activities. HCI 2012, 26th BCS Conference on Human Computer Interaction, Sept 2012, Birmingham, United Kingdom. <https://enac.hal.science/hal-01022318/>
- [HS21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.
- [HH86] E. L. Hutchins, J. D. Hollan and D. A. Norman. Direct manipulation interfaces. In D. A. Norman and S. W. Draper, editors, *User Centered System Design*, pages 87–124. Lawrence Erlbaum Associates, Hillsdale, NJ, January, 1986.
- [ID25] Interaction Design Foundation (2025). Mobile First. Accessed 23 Jan 2025. <https://interaction-design.org/literature/topics/mobile-first>
- [HC86] Henderson Jr, D. A., & Card, S. (1986). Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics (TOG)*, 5(3), 211-243.
- [Kl13] M. Kljun. (2013). The information fragmentation problem through dimensions of software, time and personal projects. Ph.D. dissertation, Lancaster University.
- [Kl25] M. Kljun. (2025). Task Information Collection. GitHub. Accessed 28/1/2025. <https://github.com/mkljun/task-information-collection>

- [KP06] Kortum, P., and Peres, S. C. (2006). An Exploration of the use of Complete Songs as Auditory Progress Bars. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(17), 2071-2075. <https://doi.org/10.1177/154193120605001776>
- [Ju00] W. S. Junk, "The Dynamic Balance Between Cost, Schedule, Features, and Quality in Software Development Projects", Computer Science Dept., University of Idaho, SEPM-001, April 2000.
- [LF24] Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., ... & Xu, Z. (2024). Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.
- [MG08] Mark, G., Gudith, D., & Klocke, U. (2008). The cost of interrupted work: more speed and stress. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 107-110).
- [MT25] Claudio Antares Mezzina ; Francesco Tiezzi ; Nobuko Yoshida - Checkpoint-based rollback recovery in session programming. *Logical Methods in Computer Science*, January 10, 2025, Volume 21, Issue 1. DOI:10.46298/lmcs-21(1:2)2025
- [MV16] Miraglia, A., Vogt, D., Bos, H., Tanenbaum, A., & Giuffrida, C. (2016, October). Peeking into the past: Efficient checkpoint-assisted time-traveling debugging. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 455-466). IEEE.
- [MD14] A. Morgan, A. Dix, M. Phillips and C. House (2014). Blue sky thinking meets green field usability: can mobile internet software engineering bridge the rural divide? . *Local Economy, Local Economy*, September–November 2014. 29(6–7):750–761. (Published online August 21, 2014). doi: 10.1177/0269094214548399
- [Nx25] Next.js (2025). The React Framework for the Web. Vercel. Accessed 23 Jan 2025. <https://nextjs.org/>
- [Ni94] J. Nielsen. Heuristic evaluation. In *Usability Inspection Methods*. John Wiley, New York, 1994.
- [No90] D. Norman. *The Design of Everyday Things*. Doubleday, New York, 1990.
- [OB24] Isabel O'Brien (2024). Data center emissions probably 662% higher than big tech claims. Can it keep up the ruse? *The Guardian*, 15 Sep 2024. <https://www.theguardian.com/technology/2024/sep/15/data-center-gas-emissions-tech>
- [OY14] M. Ohtsubo and K. Yoshida, "How does Shape of Progress Bar Effect on Time Evaluation," 2014 International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italy, 2014, pp. 316-319, DOI:10.1109/INCoS.2014.85.
- [PR15] Martin Pielot and Luz Rello. 2015. The Do Not Disturb Challenge: A Day Without Notifications. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 1761–1766. <https://doi.org/10.1145/2702613.2732704>
- [PB95] James S. Plank, Micah Beck, Gerry Kingsley, and Kai Li. 1995. Libckpt: transparent checkpointing under Unix. In *Proceedings of the USENIX 1995 Technical Conference Proceedings (TCO'95)*. USENIX Association, USA, 18.
- [PT03] Milos Prvulovic and Josep Torrellas. 2003. ReEnact: using thread-level speculation mechanisms to debug data races in multithreaded codes. In *Proceedings of the 30th annual international symposium on Computer architecture (ISCA '03)*. Association for Computing Machinery, New York, NY, USA, 110–121. <https://doi.org/10.1145/859618.859632>
- [RJ24] Imran Rahman-Jones (2024). AI drives 48% increase in Google emissions. *BBC News*, 3 July 2024. <https://www.bbc.co.uk/news/articles/c51yvz51k2xo>
- [RD02] D. Ramduny-Ellis and A. Dix (2002). Impedance Matching: When you need to know What. *Proceedings of HCI2002*. Faulkner X., Finlay J. and Detienne F. (eds), London, UK, Springer, pp 121 - 137. <https://www.alandix.com/academic/papers/GtK-HCI2002/>
- [Re25] React (2025). React: The library for web and native user interfaces. Meta Open Source. Accessed 23 Jan 2025. <https://react.dev/>
- [RP24] Gaëlle Richer, Alexis Pister, Moataz Abdelaal, Jean-Daniel Fekete, Michael Sedlmair, and Daniel Weiskopf (2024). Scalability in Visualization. *IEEE Transactions on Visualization and Computer Graphics* 30, 7 (July 2024), 3314–3330. DOI:10.1109/TVCG.2022.3231230
- [RH94] Mark Rouncefield, John A. Hughes, Tom Rodden, and Stephen Viller. 1994. Working with “constant interruption”: CSCW and the small office. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work (CSCW '94)*. Association for Computing Machinery, New York, NY, USA, 275–286. DOI:10.1145/192844.193028

- [SH14] Alireza Sahami Shirazi, Niels Henze, Tilman Dingler, Martin Pielot, Dominik Weber, and Albrecht Schmidt. 2014. Large-scale assessment of mobile notifications. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). Association for Computing Machinery, New York, NY, USA, 3055–3064. <https://doi.org/10.1145/2556288.2557189>
- [SP16] Sclater, N., Peasgood, A., & Mullan, J. (2016). Case study A: Traffic lights and interventions: Signals at Purdue University. JISC Involve. <https://analytics.jiscinvolve.org/wp/files/2016/04/CASE-STUDY-A-Purdue-University.pdf>
- [Sh82] B. Shneiderman. The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology*, 1(3):237–56, 1982.
- [Sh84] Shneiderman, B., Response time and display rate in human performance with computers, *ACM computing surveys*, Vol. 16, No. 3, 265-286, Sept 1984.
- [Sh87] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, New York, 1987.
- [SF15] Will Simm, Maria Angela Ferrario, Adrian Friday, Peter Newman, Stephen Forshaw, Mike Hazas, Alan Dix, (2015). Tiree Energy Pulse: Exploring Renewable Energy Forecasts on the Edge of the Grid. CHI'2015, Seoul, S. Korea, April 2015. ACM pp.1965-1974. DOI:10.1145/2702123.2702285
- [Si56] Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological review*, 63(2), 129. DOI:10.1037/h0042769
- [TA11] Tanes, Z., Arnold, K. E., King, A. S., & Remnet, M. A. (2011). Using Signals for appropriate feedback: Perceptions and practices. *Computers & Education*, 57(4), 2414-2422. DOI: 10.1016/j.compedu.2011.05.016
- [TG25] TANGO Project (2025). TANGO: A synergistic approach to human-machine decision making. Accessed 27/1/2025. <https://tango-horizon.eu/>
- [WK80] Walker, B. J., Kemmerer, R. A., & Popek, G. J. (1980). Specification and verification of the UCLA Unix security kernel. *Communications of the ACM*, 23(2), 118-131.
- [WH95] Yi-Min Wang, Yennun Huang, Kiem-Phong Vo, Pe-Yu Chung, and C. Kintala. 1995. Checkpointing and Its Applications. In Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing (FTCS '95). IEEE Computer Society, USA, 22.
- [WP23] Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. arXiv preprint arXiv:2303.00980, 2023.
- [WG13] Welsh Government (2013). The impact of ICT on pupils' learning in primary schools - July 2013. <https://dera.ioe.ac.uk/id/eprint/18425/>
- [WHO24] WHO (2024). Electronic waste (e-waste). World Health Organisation, 1 October 2024. [https://www.who.int/news-room/fact-sheets/detail/electronic-waste-\(e-waste\)](https://www.who.int/news-room/fact-sheets/detail/electronic-waste-(e-waste))