Hidden Figures: Architectural challenges to expose parameters lost in code

Alan Dix[1] [0000−0002−5242−7693]

[1] Computational Foundry, Swansea University, Wales
[2] Cardiff Metropolitan University, Wales alan@hcibook.com
http://alandix.com/

Abstract. Many critical user interaction design decisions are made in the heat of detailed development. These include simple parameter choices or more complex weightings in intelligent algorithms. Many would be appropriate for expert design review, user-preference choices or optimisation by machine learning, but they are buried deep in the code. Although the developer may realise this potential, the location of the decision is far removed in the code from where user feedback occurs, data can be collected and machine learning could be applied. This position paper describes several case studies and use them to frame an architectural challenge for tools and infrastructure to uncover these hidden variables to make them available for machine learning and user inspection.

Keywords: Intelligent interfaces, machine learning, user interface architecture.

1 Motivation

Many critical user interaction design decisions are not made by user experience designers informed by extensive user research, but rather by a developer working to a deadline needing to make a choice in order to progress. Sometimes these are simple parameters such as the time-delay and pixel-precision for a double click or the default choice in a menu. Some are more complex such as sets of weightings within intelligent algorithms. Many would be appropriate for expert design review, user-preference choices or optimisation by machine learning. Some are identified and maybe tuned using techniques such as A–B testing. However, often these parameters, values and decisions are buried deep in the code.

Consider the following code fragment:

```
ON mouseup
  IF ABS( mousedown_x - current_x ) <= 2
           AND ABS( mousedown_y - current_y ) <= 2
  THEN TRIGGER click
```

This is configured to allow a maximum 2 pixel drift between mouse-down and mouse up for it to be considered a click rather than a move/drag. There will be similar constants for the number of milliseconds between two clicks for them to be regarded as a double click, or in a touch-based interface for press vs hold.

It is often said that in code the only fixed numeric values should be 0 and 1, any other parameters should be in some sort of constant declaration or configuration. Equally where there is an enumeration, any use of a literal enumeration value suggests a configuration setting, and a good coder might do this:

```
CONSTANT  click_pixel_precision = 2
CONSTANT  double_click_delay    = 500
CONSTANT  default_font          = "Times Roman"
```

Imagine if these fixed values could be semi-automatically exposed in a user-preferences dialog and/or made available for machine learning.

In fact, many systems do offer the ability to tune the double-click period in user preferences, whereas the author has never encountered one that allows tuning of the pixel threshold for a click. Indeed, the latter is often zero, no movement allowed, which is particularly problematic for older or younger users, and clearly not being made available to automated tuning either.

Whether these parameters are escalated to user preferences or automated optimisation depends partly on the developer recognising their importance, but also the difficulty of achieving this in most architectures. In the case of user preferences, the path is slightly easier. The relevant configuration parameters can be made into variables in some form of central context or configuration object and set in a preferences dialog. Often this is easier to say than to achieve as the ownership both organisationally and in terms of code encapsulation may get in the way.

For automated analysis the path is far harder as the point of use needs to be instrumented, data collected, stored, passed into some form of continual or periodic batch learning algorithm and then made available for future execution of the code.

In principle it is possible to do this. However, every infrastructure, notation, toolkit and architectural paradigm has a ‘grain’, just like the grain of wood in carpentry, [7]. It is may be possible to work across the grain, but far more likely that developers will do the easy thing.

It has long been understood that the paths of connection in a user interface often cut across the module and encapsulation boundaries of functionally-defined architectures; hence the emergence and importance of frameworks such as MVC [8] or PAC [2]. More recently the availability of toolkits such as React (https://react.dev/ and Angular (https://angular.io/) has meant that real-time interactive editing of backend data has become common, despite the inherent complexity, as this has been managed within the frameworks.

Similarly, if we want good AI in UX, we need the right frameworks to make this possible.

In the rest of this paper, we will look at three case studies of system that are to some extent ‘intelligent’, but within each lie buried parameters and values which could be amenable to machine learning to tune and improve the user experience. In terms of the quadrants of AI for HCI (Fig. 1), this work lies towards to the bottom right, that is HCI applied to help developers of AI systems, but where the ultimate goal of the AI development is in the upper quadrant, short and long-term interactions incorporating AI.

This paper presents an open problem for the community, with hints of a first direction, but no solution. the author hopes that this will inspire other researchers to identify similar patterns in their own work and maybe eventually work towards systematic and generic architectural paradigms for AI-rich interactions.

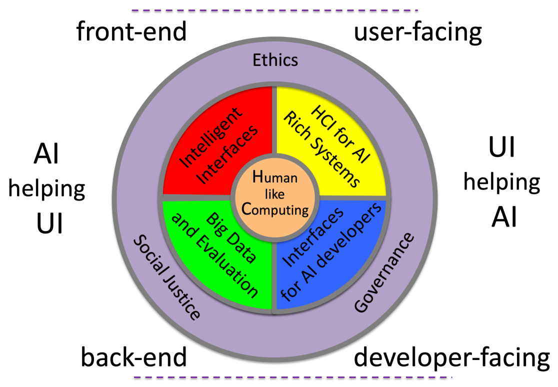


Fig. 1. The AI for HCI quadrants.

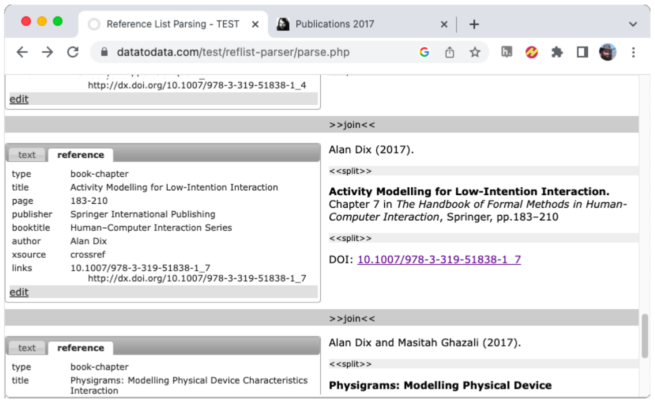
2 Case studies

These issues are not theoretical. Here are three case studies that the author has been directly involved in where they have arisen in practice.

2.1 Reading list parsing

Figure. 2 shows a prototype that was produced some years ago, when the author was working for Talis, which creates reading list software for higher education. The reading list parser makes initial best guesses at which lines to combine to make a single reference and then uses Crossref (https://www.crossref.org/) to produce structured reference data for the resulting blocks. The user can also fine-tune chosen blocks using the 'split' and 'join' separators and also edit the returned references. Crossref returns highly accurate references, but avoids parsing as it effectively uses free-text search over structured bibliographic data [9].

When the system was first created Crossref included few books and non-journal articles, and so other services were used to supplement it. A separate



Reference List Parsing - TEST

Publications 2017

datatodata.com/test/reflist-parser/parse.php

http://dx.doi.org/10.1007/978-3-319-51838-1_4

edit

>>join<<

text	reference
type	book-chapter
title	Activity Modelling for Low-Intention Interaction
page	183-210
publisher	Springer International Publishing
booktitle	Human–Computer Interaction Series
author	Alan Dix
xsource	crossref
links	10.1007/978-3-319-51838-1_7 http://dx.doi.org/10.1007/978-3-319-51838-1_7

edit

Alan Dix (2017).

<<split>>

Activity Modelling for Low-Intention Interaction. Chapter 7 in *The Handbook of Formal Methods in Human-Computer Interaction*, Springer, pp.183–210

<<split>>

DOI: 10.1007/978-3-319-51838-1_7

>>join<<

text	reference
type	book-chapter
title	Physigrams: Modelling Physical Device Characteristics Interaction

Alan Dix and Masitah Ghazali (2017).

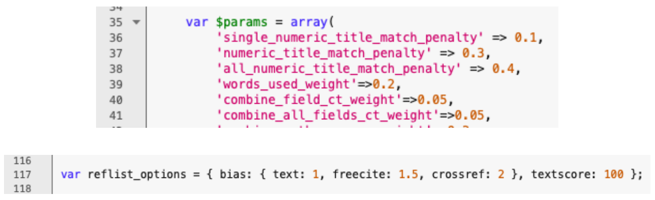
<<split>>

Physigrams: Modelling Physical Device

Fig. 2. Reference list parser: right-hand side is raw text; left-hand side parsed references.

book search was implemented using Open Library data (https://openlibrary.org/) and and Apache Solr free text index [10]. In addition, Freecite API (now retired) was used to parse the raw text references [1].

There could be results from more than one service, and some services, including Crossref, may return several responses, so the responses need to be ranked. To do this each result was assigned a relevance score. A number of metrics were used including whether the first author was in the free text reference, whether the words of the text fields in the structure reference included most of the words in the text, etc. Some metrics had internal parameters and there were also preferences between the various sources (see Fig. 3). However these were always 'best guesses' and never tuned even though the user's choices to override system selections could in principle have been used as feedback for machine learning.



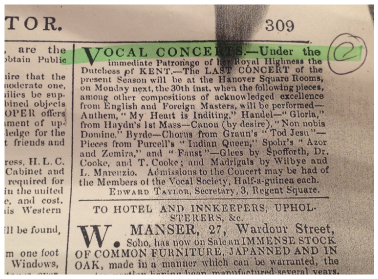
```
    var $params = array(
        'single_numeric_title_match_penalty' => 0.1,
        'numeric_title_match_penalty' => 0.3,
        'all_numeric_title_match_penalty' => 0.4,
        'words_used_weight'=>0.2,
        'combine_field_ct_weight'=>0.05,
        'combine_all_fields_ct_weight'=>0.05,
```

```
var reflist_options = { bias: { text: 1, freecite: 1.5, crossref: 2 }, textscore: 100 };
```

Fig. 3. Reference list parser parameters: server side above (fragment), client side below

2.2 Matching historic records

In the InConcert project a few years ago the author was working with musicologists looking at London concerts from the middle of the 18th Century to the beginning of the 20th Century. For more recent events concert programmes are often extant, but for the 19th Century the main source of information is concert notices in newspapers, as seen in Figure 4.



TOR. 309

, are the
btain Public

ire that the
noderate one,
ilies be sup-
bined objects
OPER offers
ment of up-
ledge for the
t friends and

ress, H. L. C.
Cabinet and
required for
in the united
e, and cost.
is Western

ll be found,

m one foot
Windows,

VOCAL CONCERTS.—Under the immediate Patronage of her Royal Highness the Dutchess of KENT.—The LAST CONCERT of the present Season will be at the Hanover Square Rooms, on Monday next, the 30th inst. when the following pieces, among other compositions of acknowledged excellence from English and Foreign Masters, will be performed—Anthem, " My Heart is Inditing," Handel—" Gloria," from Haydn's 1st Mass—Canon (by desire), "Non nobis Domine," Byrde—Chorus from Graun's " Tod Jesu"—Pieces from Purcell's " Indian Queen," Spohr's " Azor and Zemira," and " Faust "—Glees by Spofforth, Dr. Cooke, and T. Cooke; and Madrigals by Wilbye and L. Marenzio. Admissions to the Concert may be had of the Members of the Vocal Society, Half-a-guinea each.
EDWARD TAYLOR, Secretary, 3, Regent Square.

TO HOTEL AND INNKEEPERS, UPHOLSTERERS, &c.

W. MANSER, 27, Wardour Street, Soho, has now on Sale an IMMENSE STOCK OF COMMON FURNITURE, JAPANNED AND IN OAK, made in a manner which can be warranted, the

Fig. 4. Concert notice in 19th Cnetury newspaper.

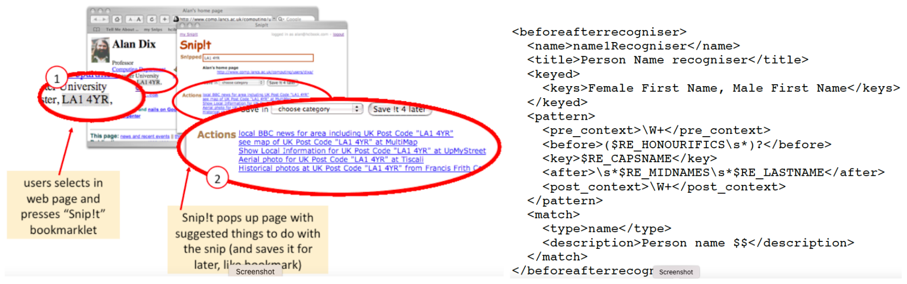
The data had been hand transcribed from the notices in a semi-structured form, but needed substantial interpretative work in order to make it usable for scholarly analysis. This included matching named entities (people and venues) within a dataset and between different datasets and also matching of different concert notices that refer to the same actual concert. Both kinds of matching needs to be fuzzy. For names people may use initials or full names and have different honorifics and for places there mat be variant spellings and descriptions. Events are more complex as there may be some or all of location, date, time, concert title, and there may be variations in many including of course that the venues may only fuzzily match.

Automatic matches were made and ranked for the musicologists to hand-check. The aim was to get the automatic matching as good as possible, but also not to miss potential matches [6, 4]. As in the case of the reading list parser, there were many parameters and weights to combine individual features, such as matched dates and levels of partially matched concert titles, to a single score. Again these were 'best guesses' and never tuned even though in the next stage the musicologists would accept or reject matches which would, in principle, give good feedback for machine learning of the parameters.

2.3 Data detectors

The final example is from Snip!t, an experimental web service that was somewhere between a web scrapbook and extended bookmark service. It allowed segments of web pages to be snipped to save for later, but retaining their provenance link to the original web page [3]. In addition, it borrowed techniques from onCue, a dot-com era intelligent internet interface, which used data detectors (a simple form of intelligent matching) to find semantic information in unstructured text [5].

Figure 5 (left) shows Snip!it in action. A post code on a web page has been 'snipped', the system recognises this and then suggests potential things to do with a postcode. The postcode recogniser is a simple regular expression:


Alan Dix
Snip!t
1
2
Save in choose category
Save !t 4 later
users selects in web page and presses "Snip!t" bookmarklet
Snip!t pops up page with suggested things to do with the snip (and saves it for later, like bookmark)
Screenshot

Actions
local BBC news for area including UK Post Code "LA1 4YR"
see map of UK Post Code "LA1 4YR" at MultiMap
Show Local Information for UK Post Code "LA1 4YR" at UpMyStreet
Aerial photo for UK Post Code "LA1 4YR" at Tiscali
Historical photos at UK Post Code "LA1 4YR" from Francis Frith C

```
<beforeafterrecogniser>
  <name>name1Recogniser</name>
  <title>Person Name recogniser</title>
  <keyed>
    <keys>Female First Name, Male First Name</keys>
  </keyed>
  <pattern>
    <pre_context>\W+</pre_context>
    <before>($RE_HONOURIFICS\s*)?</before>
    <key>$RE_CAPSNAME</key>
    <after>\s*$RE_MIDNAMES\s*$RE_LASTNAME</after>
    <post_context>\W+</post_context>
  </pattern>
  <match>
    <type>name</type>
    <description>Person name $$</description>
  </match>
</beforeafterrecogr
```

Fig. 5. Data detectors in Snip!t: (left) web service; (right) keyed name recogniser.

```
([A-Za-z][A-Za-z0-9]{1,3})[ \t]{1,6}([0-9][A-Za-z]{2})
```

However, other recognisers are more complicated. On the right of Figure 5 is one which uses initial data-lookup of personal names derived from census records in order to trigger a recogniser which combines this with regular expressions.

The matching is a simple Yes/No, but really some matches are better than others, for example a name from the data source is more likely to be a real name than something that simply matches the pattern of an initial capital. However, the complexity of choosing and then tuning these meant it was never done.

3 Architectural challenge

As we have seen there are often rich opportunities for machine learning to optimise parameters of user interfaces. However, even when the developer is aware of this potential, the location of the decision is far removed in the code from where user feedback would occur. Code would have to be substantially reworked in order to enable the automation flows. If user interfaces are to make full use of AI potential, we need architectures and frameworks to enable this.

The examples, we have seen have a relatively similar structure. At the point the relevant parameters are noticed, they need to be marked, for example, using structured commenting so that a post-processing tool can harvest them. Once extracted, a UX designer can choose to present them in user preferences and/or mark them for automated learning. In the latter case it is also important that the uses of the relevant parameters are recorded in the code where they feed into dependent values, and also where these dependent values are presented to the user; this would enable user behaviour to be re-connected to the parameterisation for creating learning sets. The final stages of machine learning and then using this to update configuration is relatively straightforward.

Of course this is not the only pattern of AI use in user interaction, so frameworks have to open and flexible enough for a wider variety of patterns. It may even be that AI techniques can be used as part of the analysis of existing code bases in order to uncover hidden UX variables for tuning.

References

1. Brown University: Freecite – open source citation parser. Retrieved from Internet Archive 29/5/2023, https://tinyurl.com/freecite (2008)
2. Coutaz, J.: Pac, an object oriented model for dialog design. In: Human–Computer Interaction–INTERACT'87, pp. 431–436. Elsevier (1987)
3. Dix, A.: Context and action in search interfaces. In: Search Computing: Trends and Developments, p. 35–45. Springer-Verlag, Berlin, Heidelberg (2011)
4. Dix, A.: Designing user interactions with ai: servant, master or symbiosis. The AI Summit London, 22nd Sept. 2021. https://www.alandix.com/academic/talks/AI-Summit-2021-UI-with-AI/ (2021)
5. Dix, A., Beale, R., Wood, A.: Architectures to make simple visualisations using simple systems. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '00. p. 51–60. ACM (2000). https://doi.org/10.1145/345513.345250
6. Dix, A., Cowgill, R., Bashford, C., McVeigh, S., Ridgewell, R.: Spreadsheets as user interfaces. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI'2016. pp. 192–195. ACM (2016)
7. Dix, A., Finlay, J., Abowd, G.D., Beale, R.: Design focus: Going with the grain. In: Human-computer interaction, p. 301. Pearson Education (2003)
8. Krasner, G.E., Pope, S.T., et al.: A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of object oriented programming **1**(3), 26–49 (1988)
9. Labs, C.: Resolving citations (we don't need no stinkin' parser). Dated: 29/11/2017. Accessed: 29/5/2023. (2017)
10. Smiley, D., Pugh, E., Parisa, K., Mitchell, M.: Apache Solr enterprise search server. Packt Publishing Ltd (2015)