CCS20 notification for paper 3

From: **CCS20** | ccs20@easychair.org Friday, Feb 14, 2:25 PM

To: **Geoffrey Litt** | glitt@mit.edu

Dear Geoffrey

We are pleased to inform you that your submission to the Convivial Computing Salon has been accepted. Congratulations! We look forward to our discussions at the Salon.

The Salon will be held in Porto, PT on March 24. You must register for the workshop at 2020.programming-conference.org/attending/registration
Note that early registration ends Feb 22. Please let us know ASAP if you will be unable to attend.

We will be in contact about planning your presentation.

Please find attached the reviews of your submission.

Regards,
Luke, Colin, and Jonathan

SUBMISSION: 3
TITLE: Wildcard: Spreadsheet-Driven Customization of Web Applications

---------------------- REVIEW 1 --------------------
SUBMISSION: 3
TITLE: Wildcard: Spreadsheet-Driven Customization of Web Applications
AUTHORS: Geoffrey Litt and Daniel Jackson

----------- Overall evaluation -----------
SCORE: 2 (accept)
----- TEXT:
Review of paper 3 - Geoffrey Litt and Daniel Jackson. Wildcard: Spreadsheet-Driven Customization of Web Applications

This paper is a well-conceived and well-executed sketch of a malleable software application --- an in-place, spreadsheet-oriented UI named “Wildcard” that can be injected into the interface of any suitable web application. The central worked example shows an Airbnb interface being augmented by the Wildcard sheet, injecting the ability to restore lost sorting functionality as well as the ability to sort by criteria derived from a 3rd-party API, “walkscore”. Another example shows the user’s personal calendar being invoked to produce a custom date picker within the context of an Expedia flight booking page.

The paper invokes many welcome concepts, such as bidirectional dataflow, the exposition of public (there named `universal') data structures, in-place toolchains, the ecosystem of users and malleable software. There is good evidence that the work is grounded in an appreciation of related work and thought.

The paper's grounding could be improved in two principal directions - firstly, looking downwards into the existing substrate, HTML and the DOM that enables such work to be done at all. It would be helpful if the paper acknowledges that this cannot be taken for granted, and is the result of an odd confluence of sociotechnical factors surrounding the birth of web technologies. The openness of the DOM substrate is constantly under threat from developments grounded in the dominant ideology of computer scientists, focused on information hiding and principles of least privilege. For example, witness the recent "Web Components" standard promoted by Google and blessed by the now effectively defunct W3C - this features a "Shadow DOM" described at developer.mozilla.org/en-US/docs/Web/API/ShadowRoot that may be closed to introspection. This defeats the potential for malleable tools such as Wildcard, and is an important ideological issue that needs to be stressed as pa!
rt of their design. A useful reference for such "anti-insulation" thinking is "Tracing a Paradigm for Externalization" presented at an earlier Salon: www.shift-society.org/salon/papers/2017/revised/externalization.pdf. Note that even the naming of crucial enabling tools such as "TamperMonkey" are done with deference to the dominant ideology that these enable misuse, rather than use, and such extensions are not supported on Google's Chrome on Android.

Secondly, the paper would benefit from a more thorough working-out of what would be required to bring about the vision of a dynamic medium on the web from this starting point. The conclusion has some good speculation about how the ecology of site adapters might be curated, and section 4.3 acknowledges that there must be "design for an ecosystem of users", but there is little critique in the paper of the "app" model of software development as a whole, or recognition that in practice we need to find ways to fold the ecosystems of users and the ecosystems of builders together. Whilst it is likely that communities based around large, fairly slow-moving and well-attested platforms such as Gmail, Airbnb and the like may be able to pay their costs in terms of maintaining adaptors for some limited tasks, it seems clear that establishing a widespread computational literacy as limned by diSessa will require a large-scale disruption of ontology of software development in order !
to be economic. This will require attention to issues such as how we establish coordinates in order to address user designs, and most importantly, how we allow them to be combined together - in the context of this paper, how tools such as WildCard themselves get developed and distributed, and how we arbitrate the situation where a user is attempting to marshal several such tools simultaneously targetted at the same environment ("web page"). Another earlier Salon paper www.shift-society.org/salon/papers/2018/revised/anatomy-of-interaction-authorversion.pdf has a small amount of exposition about the implications of connecting ecologies of function and fabrication in its section 2.

There are a few loose joints in the structure around section 4.1. Whilst this reviewer fully approves of exposed data structures, there is a quibble with the term 'universal', implying that these might be fit for any purpose. In practice, they may be only fit within a restricted ecosystem, and require networks of translation in order to be intelligible outside - they are not truly 'universal', just a form of adequately public currency. The authors cite Beaudouin-Lafon and Mackay [2000] in support of this, but the particular examples cited, of a repurposable color picker, and "polymorphic instruments" doesn't appear in that paper, which instead focuses on the notion of "polymorphic

commands" - it is a stretch to see this material lending support for externalised designs. This reviewer actually considers the notion of an `instrument' as suspicious and not of a piece with the ideas of diSessa which aim towards more-or-less homogenous "computational substra!
tes" in which agents and targets can be discovered opportunistically. The key virtue of the interchange formats described in section 4.1, that of UNIX text streams and (simple, 2-d) spreadsheets is that they are relatively unstructured. Other than the basic encoding frame which enables the information to be intelligible at all, the ontology of the contained data can be established opportunistically, and possibly also pluralistically.

The paper mentions that a drawback is "there is no way for users to express imperative workflows with sequences of actions in Wildcard" but this isn't necessarily to be considered a significant drawback. Interactions which can be expressed in "integral" terms (in the sense of the 2018 Salon paper, section 2.2) are naturally more amenable to end-user interaction and collaboration amongst multiple authors, and so it is natural to focus attention on supporting these as richly as possible before turning to the remaining hard core of inherently sequential, state-ridden interactions.

----------------------- REVIEW 2 ---------------------
SUBMISSION: 3
TITLE: Wildcard: Spreadsheet-Driven Customization of Web Applications
AUTHORS: Geoffrey Litt and Daniel Jackson

----------- Overall evaluation -----------
SCORE: 3 (strong accept)
----- TEXT:
This paper presents Wildcard, an "end user programming" system
(more-or-less) built by adding a spreadsheet onto the side of a web
browser. Tabular data can be screen-scraped from web pages, and then
viewed, edited, or processed using the spreadsheet, the original UI,
or both together. The paper begins with a brief introduction to the
idea, shows several examples in section 2, then section 3 discusses
implementation. Section 4 attempts to extract some underlying design
principles, section 5 covers some related work, and section 6
concludes.

The best papers are the ones that present the best new ideas -- and
what is often the case with the best new ideas is that they are
simple, straightforward, obvious in retrospect: that they'd hardly
count as ideas, let alone new ideas, if anyone had had those ideas
before. The point, of course, is that no-one else has. This is one of
those papers: the idea in this paper -- surfacing application data
into an "active spreadsheet" is one of those ideas.

The paper is well written, comprehensible, and easy to follow.
Although there aren't much by way of implementation details, there is
enough, and I expect if this paper went viral there could be ten
different implementations of this idea done in a week.

In many ways this draws on the postmodern / scrap-heap programming
ideas from a while ago: the ubiquity of lynx to collect data to munge
never fails to amaze. The cleverest but if Wildcard, however, is the
bi-directional links between spreadsheet and UI - sorting a table
which the UI designer never expected to be sorted (or even
intentionally prevented sorting, as in the AirBnB example in the
paper) is powerful and useful.

My main comment on this draft of the paper is that it doesn't really
discuss any weakness of this approach: there must be some, and a more
considered paper should no doubt address them - what happens if the
screen-scraping miscues? are externally imposed manipulations like
e.g. re-sorting tables always correct and "safe"? can this type of
re-writing result in misleading displays, and thus misleading
interactions (if the back end doesn't know the front end table is
sorted differently, and the back end just sends "rent room from hotel
in row 3" won't Wildcard mean the wrong room is rented?

I also surmise there is a little more going on regarding exposed data
application structures, the explicit structures provided by HTML
tables, and the ability to essentially rewrite the implementation of
those tables rather than the underlying website front or backends
themselves.

Still: a novel idea, an interesting paper, which I expect will make
for a good presentation at the Salon.

To encourage accountability, I'm signing my reviews in 2020.
For the record, I am James Noble, kjx@ecs.vuw.ac.nz.

----------------------- REVIEW 3 ---------------------
SUBMISSION: 3
TITLE: Wildcard: Spreadsheet-Driven Customization of Web Applications
AUTHORS: Geoffrey Litt and Daniel Jackson

----------- Overall evaluation -----------
SCORE: 1 (weak accept)
----- TEXT:
There seems to be a resurgence in applying spreadsheets to other types of applications, and this paper is offering a compelling demonstration of using spreadsheets as an engine for customisation of web applications.

Wildcard is a new system for editing webpages in place by representing the data being displayed in editable spreadsheet format. The paper offers several application examples, such as manipulating AirBnB search results and a todo list app, as well as an overview of how the system is implemented, and some discussion of previous research and avenues for future work.

A couple of papers that use spreadsheets as the programming language to create visualisations and graphics come to mind that would have been interesting to see discussed here as well:

Wilde, N., & Lewis, C. (1990). Spreadsheet-based Interactive Graphics: From Prototype to Tool. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 153–160.

Pantazos, K., Kuhail, M. A., Lauesen, S., & Xu, S. (2013). uVis Studio: An Integrated Development Environment for Visualization. Proc. SPIE 8654, Visualization and Data Analysis 2013, 8654, 15–30.

M Marasoiu, D Nauck, AF Blackwell (2019). Cuscus: An End User Programming Tool for Data Visualisation. In International Symposium on End User Development (IS-EUD), Hatfield, UK.

Overall, it's a well written paper, and I enjoyed reading it - for a system description paper, it's a persuasive prototype and well presented.

I wonder though how it could be made even more stimulating. It seems that generalising and saying that when it comes to working with data, we can use spreadsheets as the underlying editable data structure, and we have ourselves and end-user programming system . What seems to be missing (not just in this paper, but more generally) is a discussion at a higher level of abstractions of what is the set of things that spreadsheets make possible and when are they not enough? Can we design another/a different programming paradigm that's as end-user-friendly as spreadsheets (or maybe even more so)?

----------------------- REVIEW 4 ---------------------
SUBMISSION: 3
TITLE: Wildcard: Spreadsheet-Driven Customization of Web Applications
AUTHORS: Geoffrey Litt and Daniel Jackson

----------- Overall evaluation -----------
SCORE: 3 (strong accept)
----- TEXT:

= Review of 'Wildcard: Spreadsheet-Driven Customization of Web Applications'

by Philip Tchernavskij

== Summary
This paper presents Wildcard, a prototype tool for customizing web interfaces by manipulating a spreadsheet bidirectionally mapped to the web page's content.
The paper walks through the capabilities of Wildcard with a set of small, realistic customization tasks.
Then the authors discuss their design principles and review related research and tools from the web ecosystem.

== Opinion
This is an excellent contribution to the convivial computing salon.
Wildcard is presented very well through use cases and video clips, and the many references to literature and existing tools indicate that the authors have thought about how Wildcard fits into multiple ongoing lines of research.
I am personally happy to see work on end-user customization tools show up at the salon, as this is a particular research interest of mine.
I look forward to discussing this work further at the salon.

== Discussion
I briefly consider some avenues for extending or refining the work on Wildcard.
These may be useful in revising the paper for eventual publication, but are mainly intended as conversation starters.

=== Adversarial customization
In considering the extended network of programmers and non-programmers who could be part of a healthy Wildcard ecosystem, the authors briefly consider what role the developers of customized apps could play.
They argue that 'first party developers can play a role in enabling customization', but they don't have to.
I think this somewhat naive, if we consider how tools such as twitter, instagram, and facebook have repeatedly been re-designed to remove features that the product owner decided were not conducive to ad engagement or increased site activity.
Twitter, as I recall, made explicit moves to try to limit the development of third party clients.
The authors themselves note that AirBnB does not help users sort rentals by price, which is presumably an intentional design decision.
This is presumably intentional.

I do not think it is a stretch to imagine that platform owners might begin to obfuscate their DOM or JSON payloads, or otherwise try to break Wildcard-like tools, if those tools became sufficiently popular.
I would be interested in discussing how one could design a customization tool to foresee and work around such antagonistic responses.
Cory Doctorow has recently written about precisely this kind of 'adversarial interoperability' as part of a series of articles on interoperability for the Electronic Frontier Foundation (Doctorow, 2019).

=== Usability and directness

The authors compare Wildcard to Beaudouin-Lafon's interaction model based on instruments, because it is an 'in-place' customization tool.
Interaction instruments were originally motivated as an extension of direct manipulation design principles that could guide designers avoid filling supposedly simple and direct interfaces with e.g. long, nested menus (Beaudouin-Lafon, 2000).
Beaudouin-Lafon argues that designers should attempt to minimize the temporal and spatial distance between the interface for performing an action and the target object and effects of the action.
In this perspective, we can point to several ways in which Wildcard is not 'in-place', since it requires users to indirect into a separate pane, figure out which row or cell represents the object of interest, and then manipulate it in possibly a different modality.
On the other hand, it is clear that the authors have made some effort to make this indirection cheap, e.g. by highlighting interface elements corresponding to selected cells.
It is fine that this paper does not contain a very close analysis of this sort, but I think it is more valuable for designers and researchers than labels like 'in-place' and 'end-user friendly'.

Eagan has developed one example of an arguably more instrumental interface customization tool, grab-and-drop toolglasses, which let users pick up and reuse interface widgets in Cocoa apps (2017).
I would be interested in a discussion of trade-offs between strictly instrumental tools such as grab-and-drop toolglasses and ones that rely on a secondary representation of the interface, such as Wildcard.

=== Malleable software
I think it would be useful for the authors to qualify in a little more detail the kind of malleability Wildcard promotes.
The authors rightly note that Wildcard is an example of a tool for customizing legacy software, rather than an environment for authoring software de novo, such as Boxer and Webstrates.
Wildcard falls in the tradition of 'black-box interface extension' tools:

"Toolkits in this tradition augment legacy apps by interfacing with them at various levels. Scotty (Eagan et al., 2011) lets programmers create additional interface elements for OSX apps by injecting code through the Cocoa UI toolkit. Interface Attachments (Olsen et al., 1999) and Façades (Stuerzlinger et al., 2006) modify features of the interface of running applications through the windowing system. Improv (Chen and Li, 2017) creates cross-device interactions in the browser by transforming input events on remote devices to simulated input events on existing interface elements. Prefab (Dixon and Fogarty, 2010) uses computer vision to recognize and modify interfaces at the pixel level.
Black-box interface extensions are powerful because they create the same sort of malleability as traditional plugins, but reduce their compatibility issues. However, their purpose is to enhance, reorganize, or integrate existing systems. Therefore they are limited by the features of the apps they target and cannot be used to build systems from the ground up, nor disassemble and repurpose parts of a system."
(Tchernavskij, 2019)

'Malleable software' is not a settled term, and I have made some effort to distinguish malleability from the broader design goals of customizable software in my thesis (Tchernavskij, 2019).
I see two key points of distinction.
Firstly, the actual techniques by which interfaces are manipulated should as far as possible be direct, giving the

experience that the interface is a material that users can move, modify, and share freely.
This means, for example, that modifying underlying source code to change app functionality is a weak form of malleability, for the same reason that a nested menu is a weak form of direct manipulation.
Secondly, malleability aims to efface the boundaries of apps, helping users split and re-combine interfaces as easily as possible.
This is exemplified well by the use of transclusion in Webstrates (Klokmose, 2015).

Therefore, 'malleable software' may not be an accurate descriptor for customization tools that can only make changes to an interface within the bounds set by the host app.
I would much appreciate a citation of my thesis in the final version of this paper, together with a closer consideration of how Wildcard relates to these design goals.

Feel free to reach out at philip@tcher.tech if you want to discuss the paper.

== References

Beaudouin-Lafon, M. (2000) Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. CHI 2000.

Doctorow, C. (2019) Adversarial Interoperability. Electronic Frontier Foundation. www.eff.org/deeplinks/2019/10/adversarial-interoperability

Eagan J.R. (2017) Grab n' Drop: User Configurable Toolglasses. INTERACT 2017.

Klokmose C. N., Eagan, J., Baader, S., Mackay, W. E., Beaudouin-Lafon, M. Webstrates: Shareable Dynamic Media. UIST 2015.

Tchernavskij, P. (2019). Designing and Programming Malleable Software. PhD thesis. (In publication)