frontpage-wact.pdf

Data Analysis as a Service: an infrastructure for storing and analysing the Internet of Things

Martin Lehmann

8th May 2015

1 Introduction

Norman's (2010, 3) opinion piece 'Natural user interfaces are not natural' begins with the following quote from Steve Ballmer, CEO of Microsoft:

> I believe we will look back on 2010 as the year we expanded beyond the mouse and keyboard, and started incorporating more natural forms of interaction such as touch, speech, gestures, handwriting, and vision – what computer scientists call the 'NUI' or natural user interface."

Following the quote, a statement is made:

> [. . .] A new world of interaction is here: The rulebooks and guidelines are being rewritten, or at least, such is the claim. And the new interactions even have a new marketing name: natural, as in "Natural User Interface."
>
> As usual, marketing rhetoric is ahead of reality.

Norman is, of course, right: using Microsoft as an example, we have seen attempts at natural user interfaces in the direction of hybrid tablet computers like the Microsoft Surface, gesture trackers like Microsoft Kinect, and voice recognition in the personal assistant Cortana. Though efforts have certainly been made, we have mostly seen that Natural User Interfaces (NUIs) are just a new way to interact with the old (but sometimes slightly different-looking) interfaces we already know. A prime example how this application does not work without thought put into optimising for the task at hand is the "flat" look of Windows called "Metro" and then "Modern UI", which was supposed to fit *all* Microsoft devices (desktops, laptops, tablets, and smartphones).

Natural User Interfaces are, indeed, a new way to interact with Graphical User Interfaces (GUIs), and not something completely new. However, as we move away from pen and mouse and into the domain of touch sensitivity, speech recognition, and several other ”natural” interface types, we must also update the GUIs to suit the task at hand and the input type of choice.

For example, a tablet with a screen surface sensitive to both the natural interface of touch and the Tangible User Interface (TUI) (Ishii & Ullmer, 1997) of stylus (pen) should immediately be able to differentiate between the two input types, because a user normally only wants to use the stylus for tasks fine-grained than what can be done just as efficiently by touching the screen with fingers: for example, sketching or annotating a document.

When it has been established that NUIs are new ways of interacting with GUIs and thus get all of the benefits (and restrictions) of the GUI's exploratory nature, we must of course learn from the GUI design and how we have interacted with them before to best design natural interaction experiences – but looking to the past is far from enough. New guidelines must be written and revised before we can determine a common understanding of how natural user interfaces should be used. After all, Norman (2010, 3) calls them *not natural*.

This report looks at an issue (task) tracker designed with guidelines and best practises for Natural User Interfaces in mind. The board consists of three simple lists: **todo**, **doing**, and **done**. The user can interact with existing tasks and drag them between the lists.

The goal of the application and this report is to detemrmine how optimising a 2D interface for the natural interface type *touch* can help boost productivity in a simple issue tracking system.

2 Short literature review

3 NUIssues

In the previous section, we had a brief look at natural user interfaces as a concept and how it manifests itself in current technology, before presenting the problem this project attempts to answer: how can designing an issue tracker for the natural user interface type *touch* boost productivity?

This section is a presentation of the issue tracker, hereinafter referred to as *NUIssues*.

3.1 Definitions

Before entering a presentation of the issue tracking system NUIssues, a common domain language must be established.

- **Issue**: The conceptual, individual task to be completed. Represented graphically by *card*s, which are moved between *swimlane*s as the task's status switches between *todo*, *doing*, and *done*.

- **Card**: A graphical representation of an issue. To be moved between *swimlane*s as the status of the issue changes.
- **Swimlane**: A vertical container for issues to indicate their status. Each issue resides within a single swimlane. NUIssues has three swimlanes along the **x** axis of the GUI: *todo*, *doing*, and *done*.
- **Index**: Each *card*'s vertical (y axis) position in a *swimlane*, and (by convention) the relative priority of the issue.

3.2 Rationale behind building an issue tracker

An issue tracker in particular was selected because almost everyone has used a similar system before, be it through clinging Post-It!s to a wall or using digital systems like Trello, Atlassian JIRA, Asana, and Ding.io. This way, the user will already have a mental model for a simple issue tracker, and all focus can be on optimising the mechanics of the application for natural input. The primary goal has been to optimise the actual board interaction for **touch** input: the *natural* way of moving physical cards around is to touch and drag them.

Importantly, an issue tracker also lends itself to a highly interactive environment where almost everything can be moved to let the user customise the layout. This seems to be optimal conditions for a small proof of concept on touch optimisation.

From a software integration perspective, it is clear that visualising something as integral to an organisation as *tasks* in a clear way that provides quick insight into the state of a project will be very helpful. Given a supportive Application Programming Interface (API), should also be a relatively simple task to integrate the application with existing issue tracking systems in order to provide a more natural interface for managing tasks.

3.3 Scope and sketch

NUIssues is a system with a 2D GUI optimised for the Natural User Interface **touch**. As is to be expected of an issue tracking system, there are no aspects of Augmented Reality (AR), Virtual Reality (VR), or Mixed Reality (MR). These are certainly interesting areas to explore in the future, but not within the scope of this project.

Figure 1 shows a simple initial mockup for the application's main functionality. For comparison, figure 2 shows the implemented, much more refined and informative result.

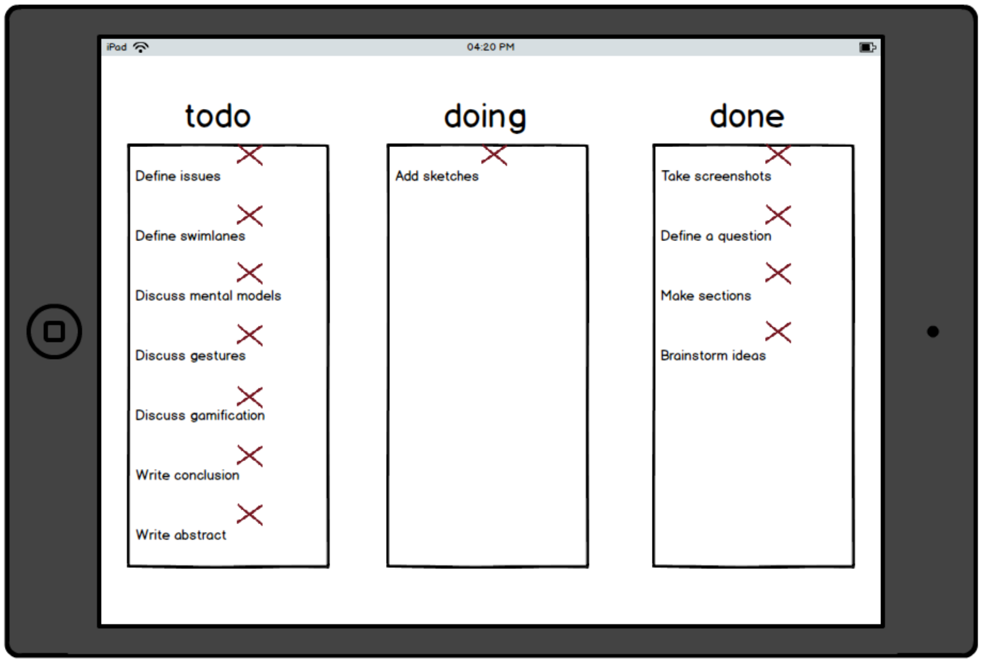

Figure 1: A simple mockup of the main functionality

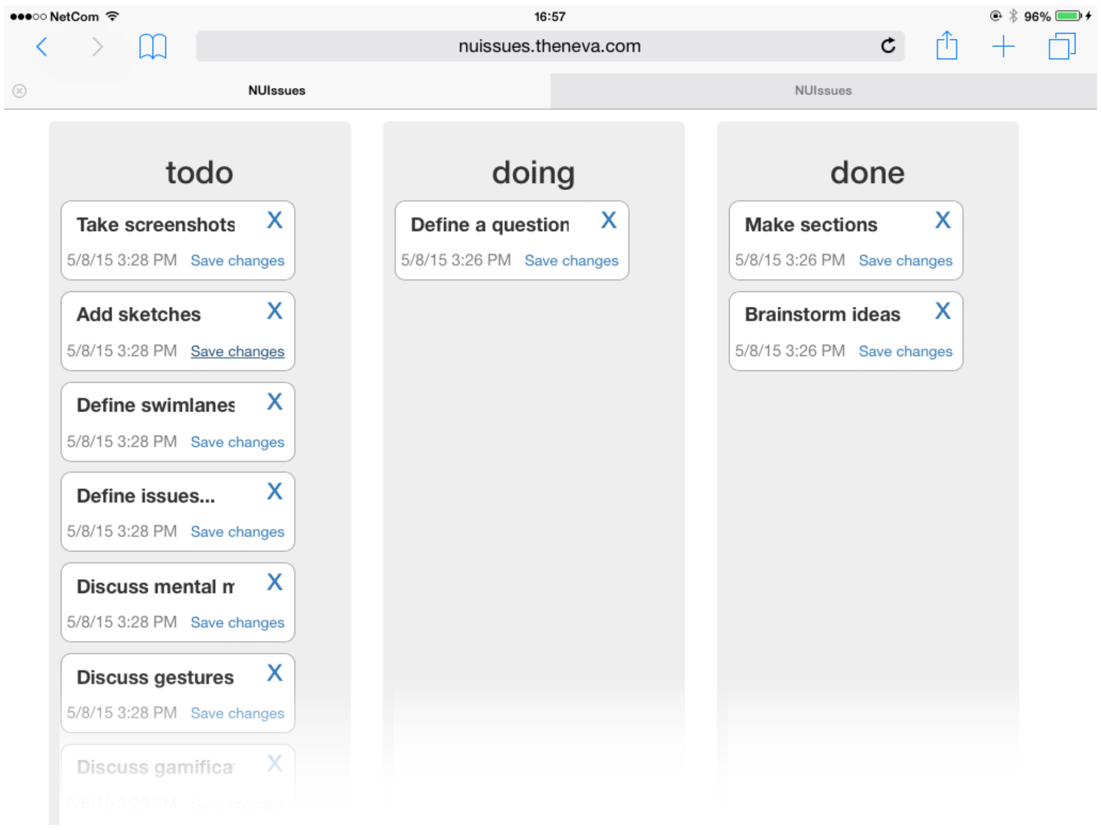
●●●○○ NetCom
16:57
96%
nuissues.theneva.com
NUIssues
NUIssues
todo
Take screenshots X
5/8/15 3:28 PM Save changes
Add sketches X
5/8/15 3:28 PM Save changes
Define swimlanes X
5/8/15 3:28 PM Save changes
Define issues... X
5/8/15 3:28 PM Save changes
Discuss mental m X
5/8/15 3:28 PM Save changes
Discuss gestures X
5/8/15 3:28 PM Save changes
Discuss gamifica X
doing
Define a question X
5/8/15 3:26 PM Save changes
done
Make sections X
5/8/15 3:26 PM Save changes
Brainstorm ideas X
5/8/15 3:26 PM Save changes

Figure 2: The system image for the main functionality

The interface is designed only for Apple iPad 1/2/Mini, but one may easily write responsive CSS to support more types of devices. This allows bringing the very simple interface to natural interfaces like *touch walls*, *touch tables*, in addition to smartphones and desktop computers. The interface is designed to be used when sitting down, although the application can certainly merely be opened if the user's intent is to grok the project's state while on the move.

If the user intends to edit textual content (only the title in NUIssues), it would be preferable to have an external keyboard. As reflected upon in the introduction, a keyboard is by far the most effective tool to write precise text in a computer system. An on-screen software keyboard will work almost as well, but overlaps almost all of the issues. Thus, the application is unusable for anything other than finishing text input and saving changes when in the editing state.

Motion controllers, gaze trackers, and brain trackers may be less interesting to use here (although Atlassian has been able to use brain power to move issues between swimlanes at a hackathon or so they say).

Could have built something native for better performance and complete control of the environment, but takes far too much time and was beside the concept we set out to prove.

Touch input: Fat fingers: no buttons (only large areas), large margins, everything can be dragged (no specific drag handles), may cause trouble with scrolling

While not explicitly expressed, one may assume that the user's model of the system identifies issues higher up (with a lower index) as more important. After all, the grid layout imposes a hierachical style on the application. This interpretation is, however, left to the user.

The system is single-user, so inter-user task coupling has not been explicitly addressed, but it is possible to evolve the system to other forms of interfaces like touch walls or touch tables (or even just real-time updates for a project).

Inter-user task coupling (irrelevant because no two users will use the system at the same time but could implement this with sockets so that both users will always know what the current status is)

Ethical problems: none in this application (except for security, duh), but certainly applicable

Gamification: no users (and no points)...

3.4 Mental models in NUIssues

Wilson and Rutherford (1989, 6) introduces the notion of *mental models* within Human-Computer Interaction (HCI), defined as any human's psychological representation (and implied expectations for) any given concept or object. A person will build a mental model for themselves, and for any object they interact with. They introduce a framework for the design phase of any computer system, with four distinctly different models at play.

First, one needs to define a *target system*. In this case, the target system is the NUIssues system and interface.

Second, there is a *conceptual model*, which is a very accurate psychological representation of exactly what the system does, usually held by an expert user or the designer themselves.

Fourth, the target system projects a *system image*, which is the implementation of the the conceptual model. This is the basis for the next element: the user's mental model of the system.

Fifth, the *user* immediately builds a mental model of the system of the system based on the *system image*. This model is the basis for what the user expects the system to be able to do, mostly based on previous experience with similar systems but only based on the hints given by the application.

Last, the *designer has a mental model of the user's model*, in which the designer attempts to guess what the user's mental model of the system image will be like. This model is built on the target audience's expected background, and can be built by considering similar applications the user can be expected to have interacted with in the past. In this case, an obvious system is that of sticky notes and checklists to track tasks, which almost everyone has used at some point. Thus, these graphical representations of issues may be used as metaphors in the system. Further, the user can have come across several digital issue trackers optimised for web like [1], Atlassian JIRA[2], Ding[3], ServiceNow[4], and IBM Notes[5].

3.5 Behaviour models

Hick-Hyman Law: about reaction time when presented with a bunch of options, not really relevant as there are zero menus (for which the law is mostly being used)

Keystroke-level model (KLM): Key stroking only relevant when adding or editing issues, pointing relevant because the finger moves, homing time minimised because of large, responsive controls, drawing minimised (and destinations hinted), mental operator, system response operator (hello Heroku)

Motor behaviour models: descriptive models: (state 0: waiting for stimuli) -¿ ¡finger down on issue handle (entire issue)¿ -¿ ((state 1: dragging issue) -¿ ¡finger up above legal area¿ -¿ (state 2: releasing "dropping" issue) -¿ (state 0: waiting for stimuli)) OR (¡finger up from text¿ -¿ (state 4: editing text) -¿ ¡tap "save changes" control¿ -¿ (state 0: waiting for stimuli))

3.6 Development process

Web client can be used by all types of devices (phones, tablets, touch tables, "smartboards", touch walls)

No specific SDKs have been used, although several frameworks are used:

- AngularJS (front-end) with ng-sortable, bootstrap 3, custom CSS
- Node.js (back-end) with several components (Express web server, Mongoose ODM (could just as easily have used relations), body-parser)
- MongoDB document database (could just as easily have used relations for this simple use case)
- Potential: JWT (or just cookies): save data inside the token

3.7 Known bugs

- It is nearly impossible to grab the bottommost issue that lies beneath the "fade out" overlay over the swimlane in a full swimlane

[1]https://trello.com
[2]https://jira.atlassian.com/secure/Dashboard.jspa
[3]https://ding.io
[4]http://www.servicenow.com/
[5]http://www-03.ibm.com/software/products/en/ibmnotes

4 Evaluation of the system in general

4.1 NUI guidelines

Direct manipulation techniques: always inline manipulation, no external prompts (except virtual keyboard on the iPad if no external keyboard is connected (which it hopefully is). Deletion option (with confirmation) on each issue.

Spatial 2D NUI guidelines: environment is optimised for touch through large elements (15mm in all directions, at least 5mm between elements on tiny tablet screen, all touch targets are equally sized), does **not** allow users to change the layout because it is not needed, only one screen -¿ simple to position controls consistently (but consistent across swimlanes), spatial relationships: higher up means higher rank/priority/importance (it is a hierarchy), todo, doing, done are obviously positioned in that order, naturally consistent, not a multi-user system (although it could have been with multi-touch recognition, which is not supported by ng-sortable but could be interesting, would not let more than one user change everyone's view, hard to indicate ownership, only uses flat/wide ”navigation” for controls, no hierarchies (e.g., dropdowns), main view has only important controls and objects, but is not too sparse

Touch input: Fat fingers: no buttons (only large areas), large margins, everything can be dragged (no specific drag handles), may cause trouble with scrolling

Sources of errors: Fat Fingers, multiple capture states, accidental activation (arm brush doing nothing to prevent this as of yet), physical manipulation constraints, stolen capture, tabletop debris (it's an iPad can't really get around that one)

Could be very cool to look at iceberg target implementation in a swimlane application

All objects on screen (except swimlanes) are draggable/sortable through land on (start drag) and lift off (release to new position) except when that means ”edit text inline”, but sliding onto a swimlane's issue list when dragging an issue causes a hint for the next free position, and sliding off the entire board when dragging an issue results in the issue not being moved

NUI gestures: not really used (should probably provide an escape sequence)

INRC: **Identity** (move issue or start changing name always does just that), **Negation** (move back, change back always undoes could look into gestures like slide to delete, slide back to restore item which is already a thing), **Reciprocal** (no undo button or anything like it, but could easily implement one), Commutative (changing name, then moving = moving, then changing name)

4.2 10 meaningful screens

Haha, 10. Each card state, hint, everything

- Swimlanes
- Single issues
- Issue that has been modified
- Issue that is marked for deletion (red: **occlusion**)
- Issue that is being moved (hint)
- Issue that is being moved (escape sequence)
- Issue that is being edited (title)

4.3 Five actionable events

Card deleted (realtime updates of the board to see what is going on?), but mostly "the time is X and we have Y tasks left to complete", or "we have way too many tasks in progress, please fix" (this falls under valuable outcome)

Potential: see who is assigned to the issue

Potential: see how long tasks are estimated to take, and prioritise based on that.

Potential for different input:

- Not using a mouse (but there's nothing in the way of using the app in a browser)
- Not using a stylus (although recognising a stylus and allowing in-place drawing of "attachments" for issues would be interesting to look into)

4.4 Valueable outcomes

4.4.1 Short-term valuable outcome

Immediately see and update the state of the project

Extremely simple interaction optimised for doing when travelling, can be used for walking, immediate overview of the entire project

4.4.2 Long-term valuable outcome

Be able to keep track of projects over time (but not across projects), be able to keep a continuous flow of issues

Focus on touch, more tactile approach, bring issues closer by letting people actually interact with them (tangible user interface?) physical form to digital information (which is really the basis of NUI in the first place, no? but not really, as there are no real objects; only digital visual representations of the actual work)

5 Conclusion and future work

Conclusion: Stuff has been done, stuff is lacking, more stuff needs to be done in stuff areas

NUIssues is a single-user system, and it would be interesting to explore multi-user issue trackers where the system for example presents several projects at the same time, and allows different teams to visually plan progress and coordinate work.

NUIssues **only** supports *touch* as natural input. While tracking changes on a user's skin and gaze tracking seem unrelated to the domain, there are several other to explore in future research: Gesture tracking, for instance, makes it possible to expand the application to run on interfaces like *touch walls*. Adding speech recognition or building a Brain Machine Interface can be relevant if the system is to be used by people who are physically incapable of interacting with a touch interface.

6 Practicalities

Link to application: http://nuissues.theneva.com (hosted on Heroku).

References

Ishii, H. & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, 411–415.

Norman, D. A. (2010). Natural user interfaces are not natural. *Interactions*, *17*, 6–10.

Wilson, J. R. & Rutherford, A. (1989). Mental models: theory and application in human factors. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *31*, 617–634.