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.

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.

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

3.1 Rationale behind building an issue tracker

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.

By definition

Type of interface: somewhere in between a GUI and a NUI (the system has a GUI which is "optimised" for the natural interface touch, built for tablets with an existing touch screen. CLI only used for development :)

NUI types: only touch (skin is unrelated to the domain, but gestures (e.g., with a kinect in a meeting room) is interesting, speech is unrelated unless the audience has no arms/ability to touch (disabled), gaze tracking is cool but unnecessary unless same as speech recognition, brain machine interface only relevant for heavily disabled users who are probably not in need of an issue tracker)

NUI in software integration (obviously integrates with a different system, and provides valuable information about the current state of the company's project(s) and thus integrates with the company as well. Very easy to set up (many companies already have iPads hanging around just get some more, or open a browser).

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).

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)

3.2 Scope

Form factor: designed only for iPad 1/2/Mini, but can easily write responsive CSS to support more types of devices (relevant for touch tables, smart boards, touch walls, smartphones). Hopefully keyboard is connected. Designed to be used when sitting down, although can be just opened (not interacted with) to grok the project's state on the move. Designed to be interacted with only when necessary and looked at when information is required

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

3.3 Sketching

Figure 1 shows a simple initial mockup for the application's main functionality.

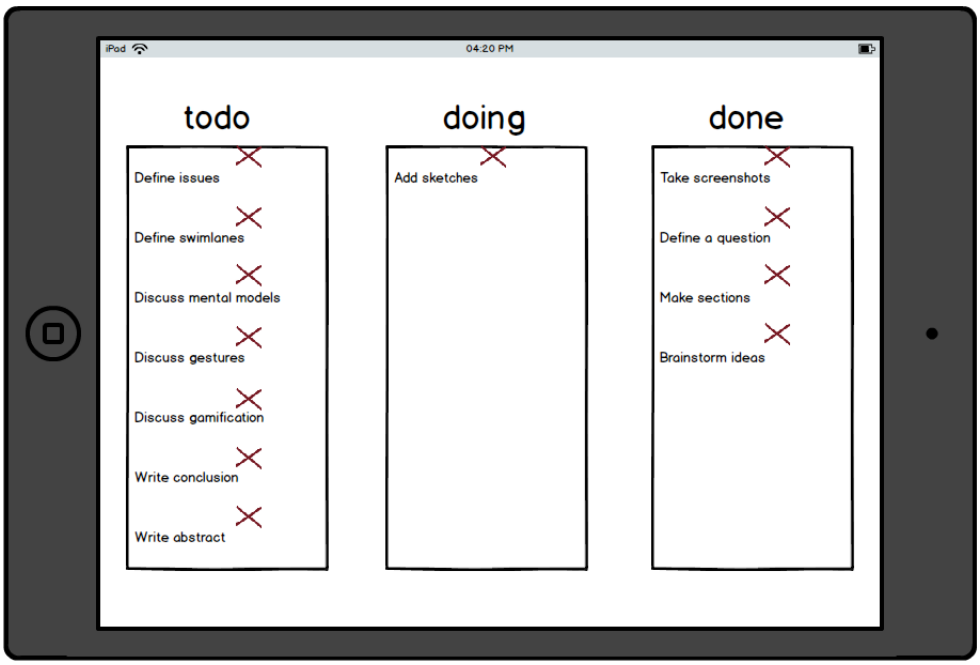

Figure 1: A simple mockup of the main functionality

Figure 2 shows the implemented, much more refined and informative result.

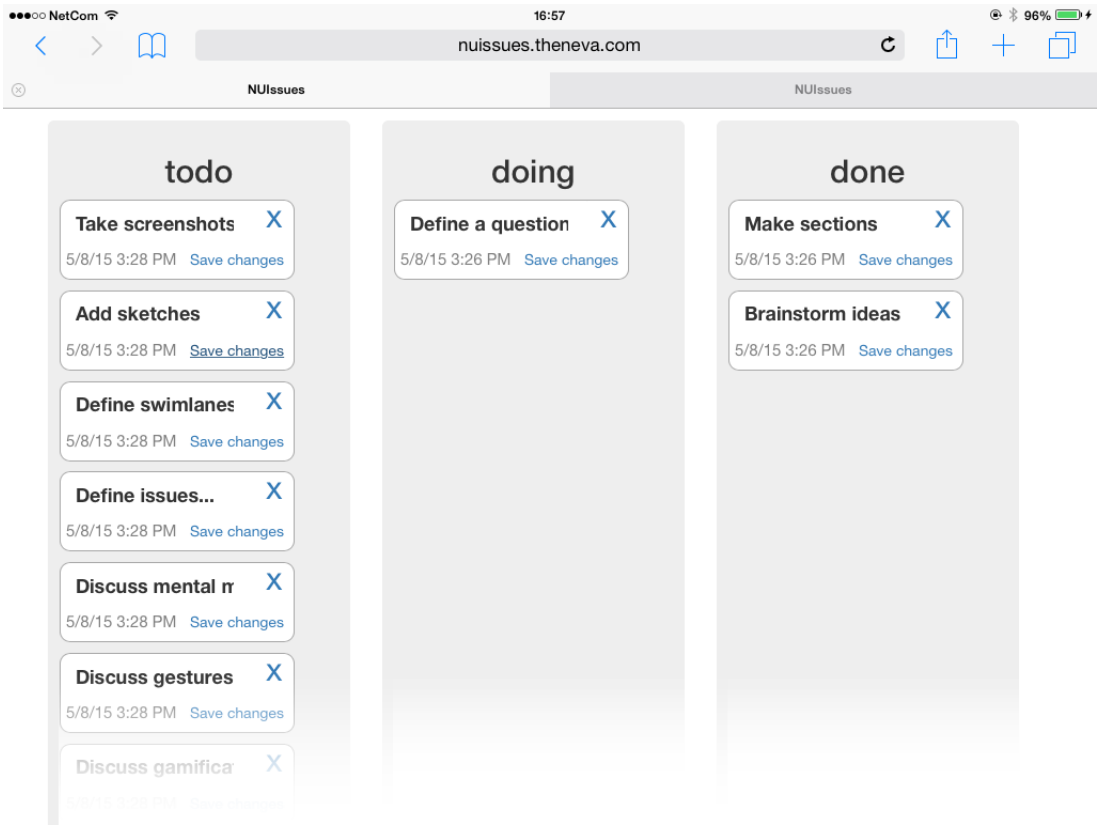

Figure 2: The system image for the main functionality

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

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

3.4 Mental models

Mental models: HCI Design: target system (duh), conceptual model (issue tracker with touch interface which supports only adding and moving/reordering issues), system image (the materialised version of the designer's mental model of the system), user's mental model of the system image (hopefully very close, but discuss this), designer's user model (hopefully very close, draw contrasts and use same metaphors/stuff as Trello, JIRA, Asana, ding.io, ServiceNow, Symfoni Notes, other issue trackers users have already used, and simple post-it note setups)

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

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

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.