Global Journal of Computer Science and Technology: G Interdisciplinary
Volume 18 Issue 1 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Klikker: A Method and Infrastructure for Mining, Analysis, and Visualisation of user Behaviour and Usability Issues for Mobile Application Development

By Mart Wetzels, Peter Peters, Idowu Ayoola, Joost Liebregts & Loe Feijs
Eindhoven University of Technology

Abstract- In the early stages of mobile application development, mockups can be used to receive feedback from potential end-users. This richness of feedback is limited by the lack of interactivity and requires a lot of time for a more significant study population. The Klikker methodology aims to unite the designer, developers, and end-users in the initial phases of development by utilising modern web technologies and readily available – and interchangeable - design and analytic software. Klikker combines the collection of quantitative user behaviour and qualitative feedback from end-users on their own devices without additional effort from researchers. The method is intended to be deployed in the first few weeks of the development process.

Keywords: usability testing, distributed systems, user behaviour, event logs, mobile applications.

GJCST-G Classification: K.6.3, H.2.8

Klikker: A Method and Infrastructure for Mining, Analysis, and Visualisation of user Behaviour and Usability Issues for Mobile Application Development

Mart Wetzels [α], Peter Peters [σ], Idowu Ayoola [ρ], Joost Liebregts [Ѡ] & Loe Feijs [¥]

Abstract- In the early stages of mobile application development, mockups can be used to receive feedback from potential end-users. This richness of feedback is limited by the lack of interactivity and requires a lot of time for a more significant study population. The Klikker methodology aims to unite the designer, developers, and end-users in the initial phases of development by utilising modern web technologies and readily available – and interchangeable - design and analytic software. Klikker combines the collection of quantitative user behaviour and qualitative feedback from end-users on their own devices without additional effort from researchers. The method is intended to be deployed in the first few weeks of the development process.

Keywords: usability testing, distributed systems, user behaviour, event logs, mobile applications.

I. Introduction

The early stages of the design and development of a mobile application often involve producing sketches of interfaces. After the ideas have been conceptualised, (interactive) mock-ups can be made using *What You See Is What You Get* (WYSIWYG)-editors such as Invision Studio (invisionapp.com), JustInMind (justinmind.com), PowerMockup (powermockup.com), ProtoIO (proto.io), Sketch (sketchapp.com), or Adobe XD CC (adobe.com). Most of these editors can produce an exported version that runs on a mobile phone and allows for testing the interaction internally within the development team. Iterations and refinements during this process depend on the experience of the team members, only at a later stage, where an actual ver sion of the mobile application is available, real user feedback can be obtained through applications such as Google Analytics (analytics.google.com), Apple Analytics (developer.apple.com/app-store/app-analytics/), AppsFlyer (apps- flyer.com), or Appsee (appsee.com).

Author α σ ρ Ѡ ¥: MSc, MSc PhD, MSc, MSc, MSc Prof Department of Industrial Design, Eindhoven Uni- versity of Technology, 5612AP Eindhoven, The Netherlands.
e-mails: m.h.wetzels@tue.nl, p.j.f.peters@tue.nl, i.b.i.ayoola@tue.nl, j.m.f.liebregts@tue.nl, l.m.f.feijs@tue.nl

The intention of the work presented in this paper aims to include (potential) users in the early phases of the development of a mobile application. Related work evaluated the benefit of performing field testing for mobile applications versus laboratory settings [Kaikkonen et al.2005]. Reviewed test procedures and tools for field testing [Bastien2010]. Provide similar remote- based user tracking methods using an HTTP Proxy [Atterer et al.2006]. WebQuilt [Hong et al.2001] has a similar strategy of visualising behaviour as a process but still utilises a separate web browser and requires the use of multiple custom services. The approach of this work is to be non-disruptive, integrating into existing workflows/software of developers, and have 'dumb' clients (no custom software or framework required).

The mining of data is achieved by attaching event listeners to existing design tools and communicating event information through web sockets to a server that also serves the interface of the mockup. The analysis and visualisation of user behaviour are treated as an Event Log used for Process Mining [van der Aalst2011]. The process analysis program Disco (Fluxicon) [Gunther and Rozinat 2012] produces a process map from the event log that enables analysis of behaviour and detects outliers in expected paths.

The asynchronous nature of recording events by the server allows for high throughput for new events. The advanced filtering methods of Disco enable removing uncompleted instances and provide a visual representation showing the bottlenecks in the mobile application design. The method discussed in this work is tested through a use-case presented in this paper on the development of a healthcare and wellbeing mobile application [Wetzels et al.2017]. The resulting application is used for the clinical trials in the European Horizon 2020 project DoCHANGE [Habibović et al.].

II. Methodology

The aim of the Klikker-method is to connect user interactions with a mock- up interface-exported in HTML format to an event log. In the next sections, 3 different points of view will be described: system view, experience view and implementation view. From a system point of view, the methodology is expressed through a system architecture model. From an experience point of view, the methodology contains guidelines to improve the user experience. From the technical perspective, a WebSocket service with a NoSQL-database (MongoDB) is used to receive and store events. Combined with a small script attached to the client-side code which monitors and sends events to that service.

a) System view

Figure 1 shows the system architecture, containing the Klikker Host, several clients, and a separate offline client. The clients connect to the web service provided by the Klikker Host, whereas the offline client receives periodical exports of the MongoDB.

The Klikker Host service runs on a local research cluster, assigned 2 CPUs, 2Gb of memory (dynamically scaling to 8Gb), and 100Gb of storage within a dedicated Virtual Machine (VM). Node.JS is used to serve the static content of the website and act as a web socket server for the clients to establish a persistent connection to. Uncomplicated Firewall (UFW) is used to restrict the in- and outgoing traffic. An SSL-certificate is used to upgrade the HTTP and WebSocket (WS) connections to HTTPS and Secure WebSocket (WSS). PM2 (Process Manager 2) is used for process management. When during peak-moments the server is on heavy use, PM2 allows quick clustering of processes. The nature of the application logic itself will enable this type of horizontal scaling without adjustments. A MongoDB instance is running on the VM; binding only a port on the localhost. Per event triggered by clients, a new document is inserted based on the type of event.

The Klikker clients consist of desktops, tablets, or mobile phones. The scalability features (user interface) from the exported website and the soft- wares ease-of-use were determining factors for choosing JustInMind (JIM) as the mock-up designing software. When loading the website, the injected script connects to the Klikker Host over a secure WebSocket connection. In addition to the source IP-address and timestamp (server-side), the con- nection is assigned a unique identifier that is used for separating events within the users (e.g. *to differentiate between users behind the same public IP-address*).

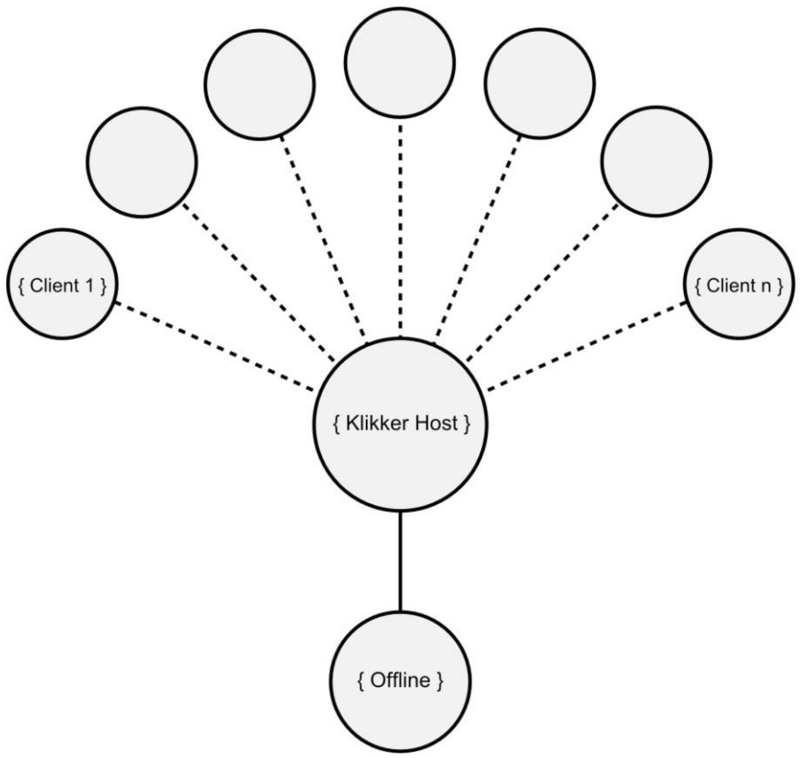

Figure 1: System architecture visualisation of Klikker method

The offline client is a desktop or laptop of the research team. The client connects to the MongoDB service through an SSH-tunnel that is configured for key-based authentication over password-based for security reasons. An export of the dataset is made using the mongo command line interface (CLI) or through the use of Studio 3T (studio3t.com). Depending on the settings of export provided by the data extraction approach, the resulting dataset will be provided as a JSON-file (JavaScript Object Notation) or CSV-file (Comma-separated Values). In our analytics setup, the JSON-file is imported into a Jupyter Notebook running a Python (2.7) kernel.

b) Experience view

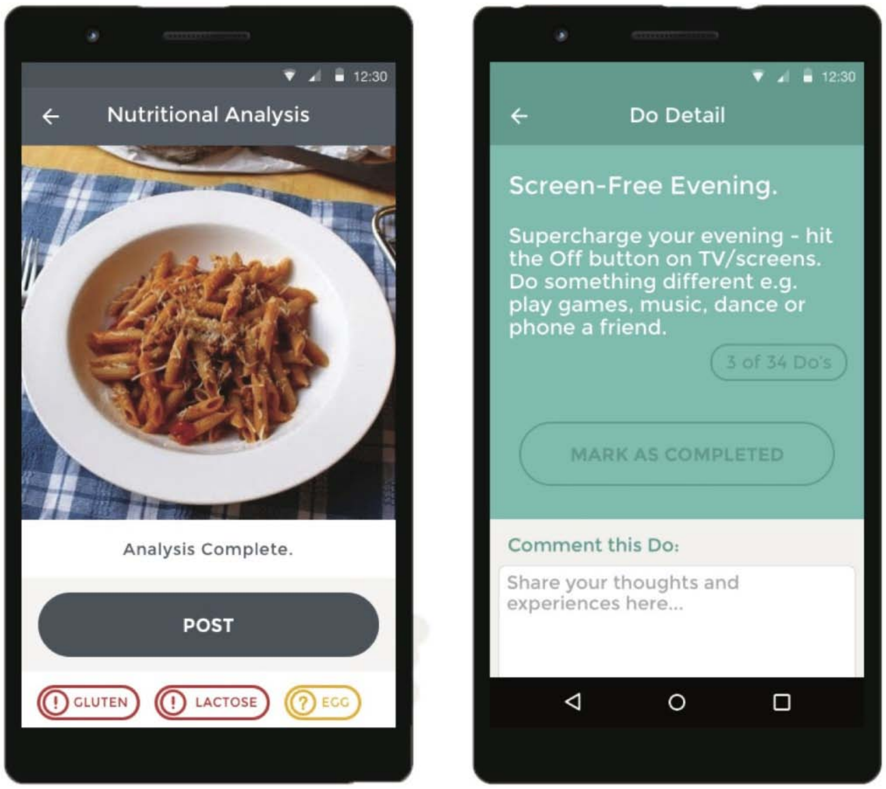
12:30
Nutritional Analysis
Analysis Complete.
POST
GLUTEN
LACTOSE
EGG
12:30
Do Detail
Screen-Free Evening.
Supercharge your evening - hit the Off button on TV/screens. Do something different e.g. play games, music, dance or phone a friend.
3 of 34 Do's
MARK AS COMPLETED
Comment this Do:
Share your thoughts and experiences here...

Figure 2: Example of interfaces used in the Klikker study

Figure 2 shows two screenshots of the DoCHANGE mobile application [Wetzels and Liebregts2017]; the use-case of this methodology. Two functionalities of the app are: being able to get insight into nutritional values of food in pictures, and receiving personalised behavioural prompts. The latter bases its personalisation parameters on data from Fitbit (fitbit.com) and Moves [Evenson and Furberg2016] using the Consume-backend [Wetzels et al.2018]. Before executing the prompts, specific to the use-case, the participant is asked to agree to informed consent and provide basic demographics, age range, and gender used to annotate the events during analysis. A feature available in JIM-exports is that when the user presses on noninteractive elements of the interface, visual cues are given to what elements are interactive. It is up to the use-case to include this behaviour, as it is monitored as well, or exclude it by changing the export settings or making the mock-up as interactive as the future product. The latter would result in more work required to create the mock-up; contradictory to the intent of the (rapid-prototyping) method.

The process of making a mock-up suitable for this purpose starts with defining what the elements that are interesting to evaluate. These elements can consist of general styling and organisation of buttons, specific functionalities, or the general structure of the app. In general, everything that can be simulated using a *Wizard of Oz* [Hannington and Martin2012] user test can be evaluated.

Although not utilised in this study, A/B-testing [Hannington and Martin2012] can be applied when hosting variations of the same mock-up. The static file serving should simply switch between versions, or a merged version is served where, based on demographics, one version is shown over the other.

c) Implementation view

Buttons and other UI elements exported by JIM are generated as HTML elements; like *div*, *img*, and *span*. The jQuery-library (jquery.com) allows event handeling on specific elements. More specifically, certain mouse-events such as *mouse-down*, *hover*, *mouse-out*, *toggle*, and *mouse-over* are available. JIM exports the mock-up inside a container with the *simulation* identifier. In case of tracking the mouse event, this prevents events being transmitted that are outside of the scope of the simulation.

The *mouse-down* event is monitored on all elements within the *simulation* container. The event handler is connected to the *simulation* element but takes the *source Element.id* of the event as an output parameter to send to the Klikker Host. This approach enables to detect participants *misclicking* - e.g. *seeing that the active area of the selection box is too small on mobile* - or clicking on non-interactive elements. The first is commonly more an issue on touch-based interfaces than on pointer-based due to the lack of granularity of the input sensors (finger versus pointer). The *mouse-down* events are used for the generation of the event log shown in the results section of this paper.

```
$('#simulation').on("mousedown",function(event)
            { socket.send({
            'event':event.srcElement.id,
            'timestamp':event.timestamp
                    })
            })
```

Code Snippet 1: mousedown event

The *mouse-move* event is also monitored on all elements within the *simulation* container. As configured in code snippet 1, The event handler is called on every refresh of the interface. According to the developer guides of Google modern devices refresh at 60 frames per second (fps); leaving about 10ms per refresh for work in each cycle [Developer]. The time consumption of extracting the event information and, most importantly, sending the web socket message can cause *janks*. Janks is a result of frame-rate dropping and results in laggy interfaces. It is advisable to reduce the collection rate or buffer events to reduce the relative time-consuming work of sending a message. In practice, sending messages over WebSockets is faster than the three-way handshake required by TCP for HTTP calls [Chaniotis et al.2015];*e.g. using HTTP calls*.

The benefit of collecting *mouse-move* events, at a reduced interval, is the ability to reproduce user behaviour as they interact with the mouse. An example of behaviour not captured by the *mouse-down* event is the indecision of a participant in choosing which button to press. Based on the *mouse-down* event, a long time span is observed between two events, but it cannot explain what the user was doing in between. With the *mouse-move* event, a researcher could differentiate between indecisiveness while moving the mouse and *Away From Keyboard* (AFK) as is possible with traditional tracking software.

```
$('#simulation').on("mousemove", function(event){
      socket.send({
      'x':event.offsetX ,
      'y':event.offsetY ,
      'timestamp':event.timeStamp
         })
      })
```

Code Snippet 2: mousemove event

Code Snippets 1 and 2 show the code required for handling these events and sending them to the Klikker Host. As can be seen, no identifiable infor- mation is stored on the client or appended to the message. The identifiers are added to the message by the server based on connection details. This approach results in a database, or *collection* for MongoDB, that consists of thousands of separate events that can be regrouped based on identifiers but are stored as independent data-points. The timestamp keys ensure reproducibility of chronological order. The combination of identifiers and timestamps result in not having to rely upon or even implement, session- management in the Klikker Host.

III. Results

This section addresses the application of Klikker-methodology on the use- case of the development of a mobile health application for the DoCHANGE project. Prior work consists of defining the functionalities and producing a visual style of the DoCHANGE application. The results section is divided into the four phases: generation, collection, analysis, and evaluation.

a) Phase 1: Generation

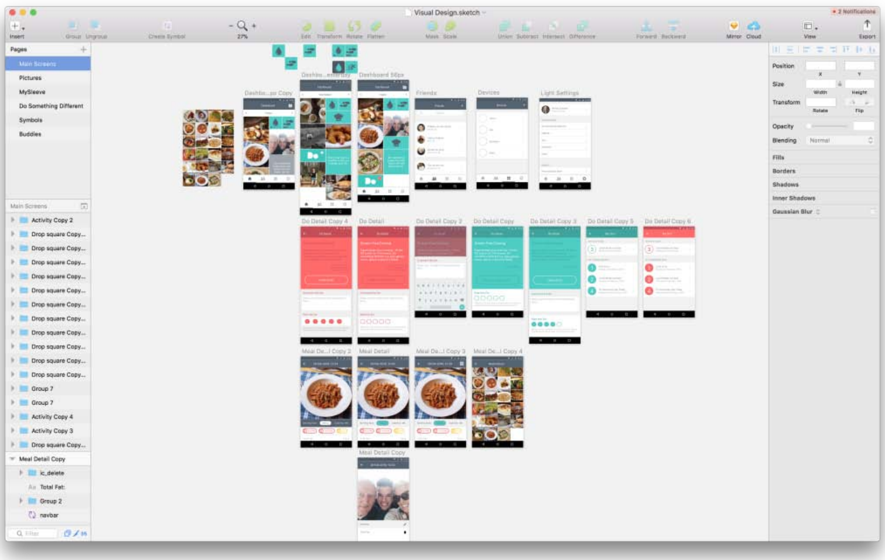

Figure 3: Overview of screens for mockup procuded in Sketch

Figure 3 shows a screenshot of the mockup design created in Sketch. The primary role of this overview is to iterate over the chosen visual language and conceptualisation of specific features. The overview should contain an exhaustive amount of screens that cover the range of possibilities for the mobile application. The application development team can define focus points for user analysis.

The DoCHANGE mobile application aggregates data from various commercial devices, such as Fitbit, and newly developed devices and services such as Smart Sleeve [Ayoola et al.2014] and Horus. This mobile application intends to preside next to existing applications from the devices. Smart Sleeve is a liquid intake monitor for heart failure patients with a fluid restriction. Horus is a service for providing nutritional advice based on photographs.

In this use-case, the following tasks are given to participants:

1. Upload a picture from the camera roll. Use a family picture for this one.
2. Keep a record of what you eat. Upload a picture of your last meal.
3. You have just received a new DO (message)! Check it out and mark it as completed.
4. Change the profile settings so buddies can find you.
5. Text one of your buddies on your Buddies list.
6. Add a new device to your Devices list.
7. Check your liquid intake data.

Based on the sequence of tasks, the interface of the application can change. *E.g. after Task 1, a family photo should be visible on the dashboard.* This type of detail is added to make the mockup experience feel complete and bind the result of a task to the next task. As depicted in Figure 4, after completing a task, a separate screen is used to introduce the next task. Pressing the start button serves a normalisation timestamp to compensate the time difference between tasks.

The interactive mockup is internally tested by the development team as a dry-run test for completeness. After exporting to an interactive website, the mockup-with the custom integration injected- is placed on a dedicated server hosting the static files. Again, the development team performs a dry-run to identify the correct functionality of the service.

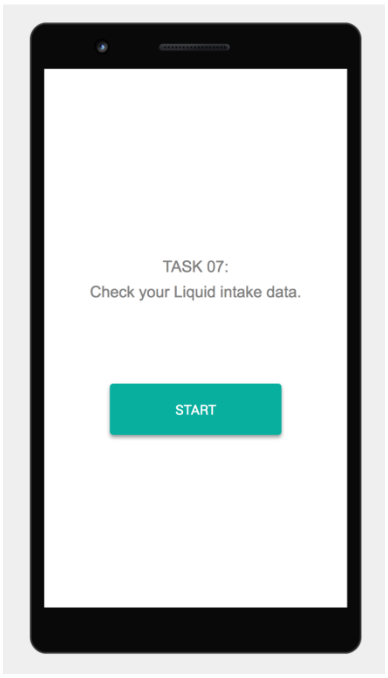

Figure 4: Example screenshot of task screens

b) Phase 2: Collection

The Klikker study for the DoCHANGE project was put online for three weeks and available online through a link. All members of the consortium were asked to distribute the link through social media or by word of mouth. The consortium partners are situated in The Netherlands, Spain, United Kingdom, and Taiwan; which explains why the data points shown later on Figure 6 clusters around this regions.

Before the in-depth analysis, a quick view of general statistics is per- formed to determine whether the size of the dataset is satisfactory for the analysis. *E.g. if the sample size is too small, additional recruitment efforts need to be performed.* Using the Mongo CLI, an export was made of the full dataset in a JSON-format. This dataset was transferred using *scp* (Secure Copy based on SSH protocol). A total of 160 unique users - based on the IP-address, connection identifier, and submission of demographics before the first task - are found in the exported database. These have produced 5097 events based on the *mouse-down* event handler described in the methodology section. For the events *sourceElement.id*, 165 unique events are found in the dataset.

For the study, gaining insight on the user behaviour and difficulties in executing the predefined tasks, it was concluded that the continuation of the analysis of the methodology was warranted. As mentioned above, in the case where more data would be needed, it would simply require recruiting more participants and export a new dataset.

c) Phase 3: Analysis

The exported dataset is imported into the Disco program. Every participant is considered as a case for the Event log and their button presses as their activities. The *sourceElement.id* values are labeled as *Activities* and an aggregation of IP-address and connection-id is used as *CaseIDs*.

An extension to the basic analysis showed a mean case duration of 2.9 minutes (median 99.5 seconds). Based on internal testing, the questionnaire can be performed in 4 minutes. This observation triggered the investigation towards the length of the cases (number of activities). A high number of cases did not finalise all tasks. For that reason, a filter was applied to take out all cases that not completed up to Task 2. Task 2 was chosen to not discriminate non-compliance to the study based on the difficulty of the task versus not wanting to finalise participation. The filter reduced the sample size to 94 participants.

The remaining cases are imported into a Jupyter Notebook (jupyter.org) for analysis. Figure 5 visualises the age-range and gender distribution of the remaining 94 participants. The Figure highlights a bias in the sample for males between the age of 18 and 45 (combining the lower age-ranges).

Based on the IP-address of the case, a geo-distribution is made using the Map View plugin (Figure 6). The Figure illustrates the bias for Dutch participants over the other partner countries. This visualisation does not take into account the possible use of a Virtual Private Network (VPN) connection, hiding the actual geo-source of the participant, and can provide a visually skewed picture based on the different sizes of the countries. *E.g. the number of participant from Finland might appear larger than those of Spain.* Although this visualisation is not a crucial element to the analysis, it does provide an insight to the participant population as the geolocation is available as a second parameter in the dataset. Considering the ease of producing such overview, it is recommended to generate it to identify any bias.

Figure 7 shows a map of general pathways of cases through the tasks. The red line highlights the correct pathway to complete the Tasks. The pathway is filtered to hide activities with relatively low occurrence for clarity of the visualisation. Deviations from the correct path show unintended user behaviour. Based on the activities that are performed, the intuition of the participant, expecting a specific action resulting in the correct outcome, can be reconstructed. As discussed in the evaluation section, this can result in concrete design- and development-requirements.

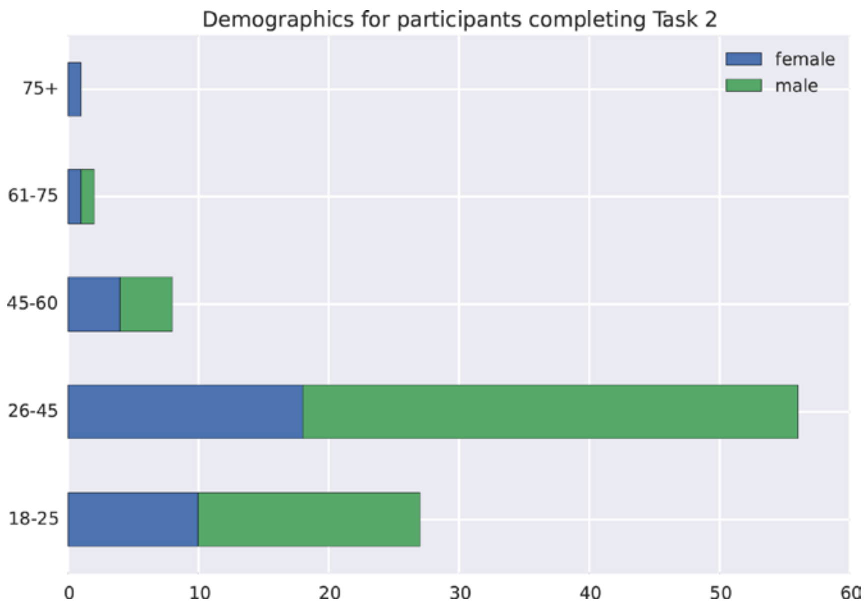

Figure 5: Stacked bar chart showing demographics of participants

After completing all tasks, the participants were given the opportunity to provide written feedback to express experiences not captured during the study. As presented in the evaluation section, 92 out of 94 participants provide additional feedback.

d) Phase 4: Evaluation

Based on the results from the analysis, several learning can be derived. Looking at Figure 7, the circles highlight the bottlenecks in tasks. A bottleneck is defined as a high deviation from the correct pathway. These bottlenecks occurred at the following tasks:

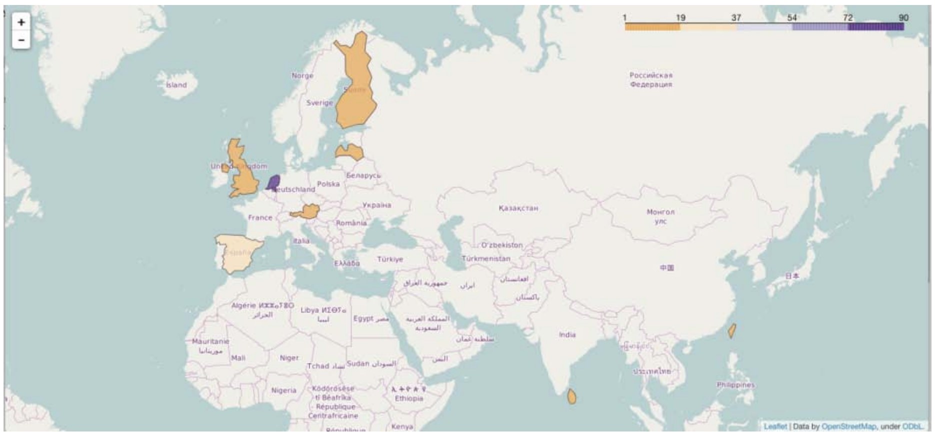

Figure 6: MapView of participant geo-distribution

Task 4: Change your proftle settings so buddies can ftnd *you* Al- though the majority followed the correct activity, going to settings to change, the second largest group searched for the settings in the buddies panel. This shows unclarity in how to change the profile settings.

Task 5: Text one of your Buddies on your Buddies list the majority chose a buddy that was not listed as a friend yet; the top of the list. To stimulate the social experience from the application, a suggested friend list is put on top of the screen. Considering that the largest group chose the first item on the list, it can be concluded that the suggested buddies disrupts the user flow.

Task 6: Add a new device to your Devices list As stated in the qualitative feedback, and low number of participants choosing the correct event, the intended device button in the interface did not identify itself as such.

Based on the qualitative feedback, the following observations are extracted from the entire list of feedback:

- The visual style of mock-up was general considered to be good. Some Tasks, involving pressing icons as buttons, created confusion to the purpose of the button. *E.g. the devices list button consisted of four squares.*
- Multiple participants stated that they only finished a specific tasks based on trial-and-error or due to the interactive element highlighting from JIM.
- The purpose of the study was not clear to some participants as no introduction of the study was provided. This was done intentionally by the researchers to prevent pre-educating the participants for the purpose of the app. If required, additional introductory stories can be added prior to loading the mock-up to provide some context to the study.
- Some participants wanted to see more of the application; this is limited by the mock-up and outside the purpose of this approach.
- Feature-specific feedback consisted of raising the concern of showing *People you might want to know* as suggestion in the Buddies list or prioritising these suggestions over actual Buddies by showing the sug- gestion first.
- These insights provided direction for future efforts: re-evaluate the application navigation and investigate how to improve the usage of icons to convey meaning. The latter might seem obvious from a third perspective, but bias blind spots are only discovered by receiving feedback from others.

IV. Conclusions

The Klikker-methodology has proven to be easy to implement and requires few resources in comparison with other methods to collect user behaviour and feedback. The process requires the involvement of researchers with different backgrounds to create the mockups and analyse the produced dataset. Prior work by the authors, also highlighted the relevance of combining the perspectives of researchers, designers, and developers [Wetzels et al.2017]. The Klikker method intends to achieve this cooperation at an early stage of the application development to increase efficiency in the development process and prevent major overhauls in the later stages of development. From a technical perspective, the authors would recommend modernising the setup by containerising all services using Docker [Chamberlain and Schommer2014], implementing a reverse proxy-such as NGINX [Nedelcu2010] or Traefik (traefik.io)-, and having NGINX or a separate service to host the static files instead of the Node.JS process. Future efforts will also investigate the mimicking of the TCP three-way handshake using web sockets to confirm successful delivery of events. From a research perspective, the method is positioned as an early stage explorative approach to kick-start an iterative development cycle and promote end-user involvement. The method has not been designed to evaluate large, in depth, topics such as user behaviour over a longer period. It is recommended to focus on UI elements and general functionalities; preferably the study does not take more than a few minutes.

Acknowledgements

This work is supported by the European Commission Horizon2020 funded Do Cardiac Health: Advanced New Generation Ecosystem (DoCHANGE) project (Grant number 643735). The authors declare no conflict of interest. The Do CHANGE team has received funding for Research and Innovation from the European Union for the project.

References Références Referencias

1. [Atterer et al. 2006] Atterer, R., Wnuk, M., and Schmidt, A. (2006). Knowing the user's every move. *Proceedings of the 15th international conference on World Wide Web - WWW '06*, page 203.
2. [Ayoola et al. 2014] Ayoola, I., Ansems, K., Chen, W., and Feijs, L. (2014). Design of a Smart Cup A Tele-medical System for Behavioral Interven- tion through Interactive Materiality. In *the International Conference on Health Informatics*, volume 42, pages 110–113.
3. [Bastien 2010] Bastien, J. M. (2010). Usability testing: a review of some methodological and technical aspects of the method. *International Journal of Medical Informatics*, 79(4):e18–e23.
4. [Chamberlain and Schommer 2014] Chamberlain, R. and Schommer, J. (2014). Using Docker to support Reproducible Research (submission to WSSSPE2). *Figshare*, pages 1–4.
5. [Chaniotis et al. 2015] Chaniotis, I. K., Kyriakou, K.-I. D., and Tselikas, N. D. (2015). Is Node.js a viable option for building modern web applications ? A performance evaluation study. *Computing*, 97(10):1023–1044.
6. [Developer] Developer, G. Web Fundamentals- Rendering Performance.

7. [Evenson and Furberg 2016] Evenson, K. and Furberg, R. (2016). Moves app: a digital diary to track physical activity and location. *British Journal of Sports Medicine*.
8. [Gunther and Rozinat 2012] Gunther, C. W. and Rozinat, A. (2012). Disco: Discover Your Processes. In *BPM 2012 Demonstration Track Demonstration Track of the 10th International Conference on*, number September, pages 40–45.
9. [Habibovi ć et al.] Habibovi ć, M., Broers, E., Piera-Jimenez, J., Wetzels, M., Ayoola, I., Denollet, J., and Widdershoven, J. Enhancing Lifestyle Change in Cardiac Patients Through the Do CHANGE System ("Do Cardiac Health: Advanced New Generation Ecosystem"): Randomized Controlled Trial Protocol. 7: 1–9.
10. [Hannington and Martin 2012] Hannington, B. and Martin, B. (2012). Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions. pages 124–125. Rockport Publishers, february 1 edition.
11. [Hong et al. 2001] Hong, J. I., Heer, J., Waterson, S., and Landay, J. a. (2001). WebQuilt: A proxy-based approach to remote web usability test- ing. *ACM Transactions on Information Systems*, 19(3):263–285.
12. [Kaikkonen et al. 2005] Kaikkonen, A., Kek äl äinen, A., Cankar, M., Kallio, T., and Kankainen, A. (2005). Usability Testing of Mobile Applications: A Comparison between Laboratory and Field Testing. *Journal of Usability Studies*, 1(1):4–16.
13. [Nedelcu 2010] Nedelcu, C. (2010). *Nginx HTTP Server: Adopt Nginx for Your Web Applications to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever*. Packt Publishing Ltd.
14. [van der Aalst 2011] van der Aalst, W. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, volume 136.
15. [Wetzels et al. 2018] Wetzels, M., Ayoola, I., Bogers, S., Peters, P., Chen, W., and Feijs, L. (2018). Consume: A privacy-preserving authorisation and authentication service for connecting with health and wellbeing APIs. *Pervasive and Mobile Computing*, 43:20–26.
16. [Wetzels et al. 2017] Wetzels, M., Liebregts, J., Ayoola, I., Peters, P., and Feijs, L. (2017). Why Healthcare and Well-being Researchers should Be-come Developers: A Case Study Using Co-Creation Methodology. In *Proceedings of the Conference on Design and Semantics of Form and Movement-Sense and Sensitivity, DeSForM 2017*.
17. [Wetzels and Liebregts 2017] Wetzels, M. and Liebregts, J. M. (2017). Vire DoCHANGE.

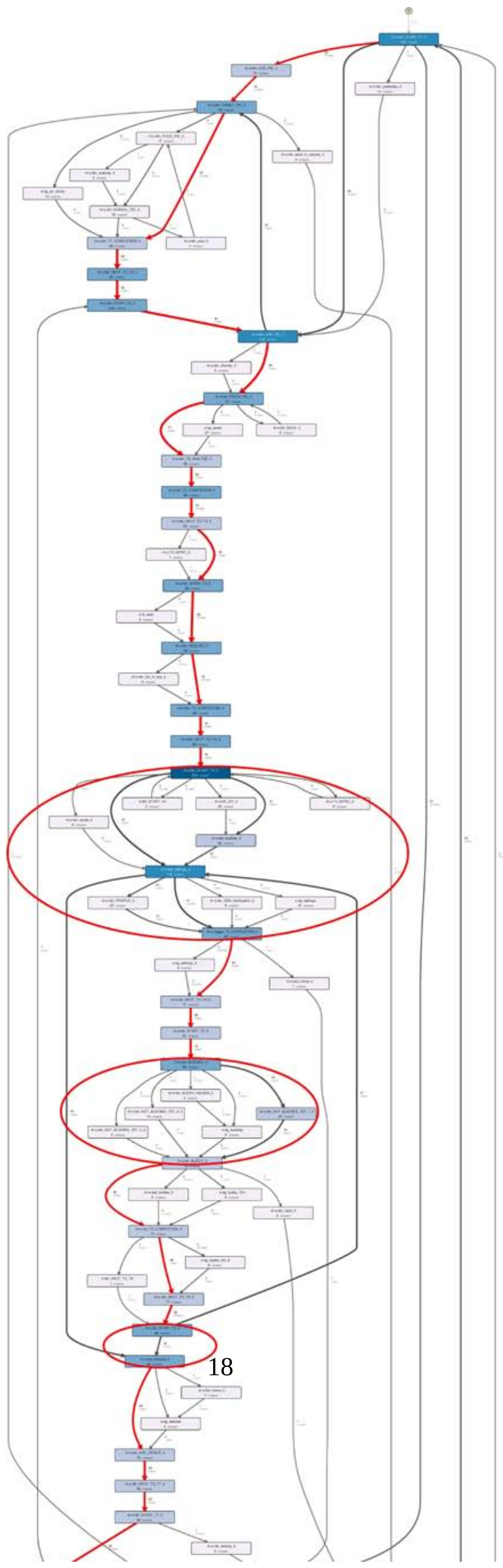

Figure 7