# Integrating a Cognitive Modelling Framework into the Design Process of Touchscreen User Interfaces

Patrick K.A. Wollner(✉), Patrick M. Langdon, and P. John Clarkson

Department of Engineering, Engineering Design Centre, University of Cambridge, Cambridge, UK
{pkaw2,pml24,pjc10}@cam.ac.uk

**Abstract.** Interface design is often constrained by the limited scope and resource-intensive nature of conventional user studies. We aim to unburden this process by introducing an automated user modelling framework that continuously injects design guidelines into the development process. We present a pipeline that converts a given user interface design into a widgetised data structure, executes a performance simulation based on the cognitive model of a user, and analyses its output to give design guidelines. We introduce the research methodology employed to create the model, implementation details of the model, and initial results from its validation. These include the dynamics of age-based modelling, the temporal integrity of the output of the cognitive model, and indications of the accuracy of the overall design guidelines produced.

**Keywords:** Inclusive design · Universal design · User interfaces · User experience · Usability testing · Cognitive modelling · Cognitive architectures

## 1 Introduction

The quality of user interface designs on touchscreen devices is often constrained by the lack of user testing on these interfaces during the developmental stage. This is the result of the incompatibility between the extremely short product cycles and the significantly greater time requirements for comprehensive and conclusive tests involving real users from a broad range of prior experiences.

In this paper, we introduce the overview of a user modelling pipeline which has the potential to be integrated into established user interface design processes without additional time affordances. Thereby, the pipeline allows access to new insights into the potential pitfalls of a given interface design proposal without creating additional friction in the process. In addition to the specifics of the pipeline and its usage, we outline how the Design Research Methodology (DRM) [3] was employed to realise the pipeline, including a brief overview of the results from its associated validation substage.

## 2 Methodology

The pipeline was designed employing an adapted version of the DRM [3], specifying the DRM Criteria to match the shortcomings of existing user modelling implementations, both in terms of their specificity to mobile touchscreen devices and the environments they are used in, as well as the quality of potential interpretations of their output. The integration of the DRM into the existing components is outlined in Fig. 1.

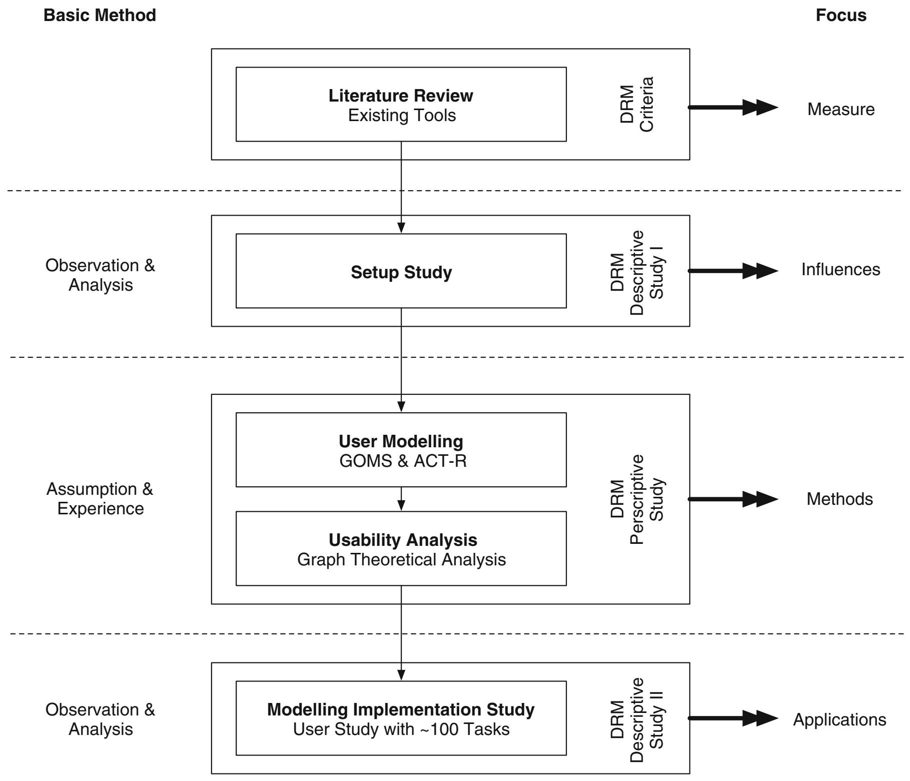

**Fig. 1.** Overview of the methodological approach

A comprehensive literature review was completed [12] to determine the DRM Criteria. In this case, the adaption of a pipeline, with the specific qualities applied to its potential for application in inclusive design, the overall applicability for touchscreen applications, and the specifics of mobile usage, signify the key research criteria.

The first Descriptive Study was completed as a pilot user study, using first-time-use tasks as an indicator of the factors that influence the aforementioned criteria. It was based on the observation and analysis of twelve users of first-time setup tasks and is outlined in Wollner et al. [10]. The focus of this work was to gather the necessary fundamental *influences* for all further work in this body of research.

Subsequently, the model was implemented utilising ACT-R [1] as the underlying cognitive architecture, combined with a GOMS-based widgetisation [5] of the elements on individual screens or screen states. According to the DRM, this was the Prescriptive Study associated with the body of the work. Based on a number of assumptions and the influences of existing work, the modelling was further extended and integrated with a usability analysis framework that employs graph theory and message passing – two interdependent approaches that integrate the density of user performance data with the sparse interconnectedness of the user interfaces tested. More specifically, the holistic analysis of the output was accomplished by employing a graph theoretical inference network, which is based on the network of the user interface states and the performance metrics generated by the cognitive model. This is outlined in Sect. 3.3.

Additionally, to impact the design process, the output of the inference network was transformed and thereby simplified to a topological representation of the screen states that could be interpreted directly by designers and developers alike, impacting the design process without generating additional friction in the process. This allowed the modelling implementation to be further refined, building the foundations of the pipeline introduced in this paper.

Finally, a validation study was completed, as outlined in Sect. 4. It is based on an analysis of the key determinants of the newly introduced methods and involved a sample of nearly 100 tasks performed by a sample of users with varying backgrounds, each performing four independent tasks on a mid-sized touchscreen device.

## 3 Pipeline and Usage

In contrast to existing tools and methods that aid the design of graphical user interface applications – particularly on touchscreen devices – we introduce an approach which consists of four distinct components. We translate the proposed design of a device into a machine-interpretable description, analyse it using a cognitive architecture (ACT-R, [1]), interpret the results by employing a graph theoretical approach, and, finally, inject the resulting insights into the design process. For each of the above steps, we have developed a comprehensive approach, which is outlined in a number of related previous publications (including [10–12]), is concatenated in the first author's thesis, and, more concisely, in this paper.

The pipeline was carefully constructed to meet the needs of modelling accuracy, design integration and design impact. As such, each of the above mentioned stages were developed on the basis of potential impact. Prior to going into more detail, the pipeline stages are outlined in the list below.

1. **Interface Representation.** The acquisition of a representation of an interface, containing the interconnectedness of all interface states as well as the widgetised visual design of the individual interface states.

2. **User Modelling.** The modelling stage in which a cognitive architecture is employed to simulate a range of user types on the interface representation.
3. **Usability Analysis.** A mathematical framework which exploits the information on the interconnectedness of the UI and the results from the User Modelling stage to approximate the anticipated interface progression of a real user.
4. **Design Injection.** An environment in which the output of the previous stages are visualised in a format that is of value to developers and designers, without the necessity of prior training and with the greatest possible impact on the design process.

## 3.1 Interface Representation

For modelling approaches to be implemented, it is necessary to translate a given design proposal into a machine-interpretable format. Utilising GOMS-variants [5] as the basis for this procedure, we read and analyse the widgets of any given design, categorise them into a predefined set of standard elements, and assign modelling classes accordingly. This descriptive process is performed in an XML format that is translated into so-called *chunks*, which, in turn, can be read by the modelling environment employed by the next step of the modelling framework.

In accomplishing this, we analysed existing GOMS implementations and extended them according to three key requirements: (i) an inclusive audience (i.e. older users or users with varying capabilities), (ii) touchscreen use and (iii) applicability in modelling mobile environments. There are a variety of GOMS implementations and extensions [4] that can be broken into four main categories and were analysed as such:

**CMN-GOMS.** A plain implementation of GOMS, written in pseudo-code first introduced by Card et al. [4].

**KLM.** Keystroke-Level modelling, as employed by GUI cognitive modelling software packages, such as CogTool [6] which acts as a simplified model/version of GOMS.

**NGOMSL.** Natural GOMS Language [8], which acts as a stricter version of GOMS, provides well-defined, structured natural language and estimates learning time.

**CPM-GOMS.** Cognitive perceptual motor analysis of activity path, this may also be described as a critical path method which is based on the parallel multi-processor stage of human information processing [7].

Within an inclusive context, it is especially important to outline the shortcomings of GOMS for highlighting the exclusion factors of the interfaces investigated with this method. Schrepp [9] outlines some of the challenges that GOMS-techniques face, with emphasis on the requirements of older users. The resulting work outlines how GOMS techniques could be "adapted to evaluate the efficiency of interface designs for [older and] disabled users" [9].

This included a particular focus on the factors of attention and repeated screen views, such as scrolling. More specifically, we defined a widgetised description of the impact of user attention in a range of scenarios on the widget in question, allowing the dependency of the part of the user interface to be defined by the impact it will suffer by limited user attention.

Likewise, a key component not covered by conventional GOMS implementations was the aspect of repeated views based on scrolling. Traditionally, elements of a user interface would only be seen once by the user, without the scope for scrolling the view. In reality – and mainly due to novel interaction techniques such as touchscreen user interfaces following the style guides of the major (mobile) operating systems such as iOS and Android – these elements have all become scrollable. As such, elements that would not be visible after an action on the interface now remain (partially) visible after the page (on which the widgets are contained) is scrolled. This is something that had to implemented in both the description of modelling and the interpretation of what GOMS segmentation of visual elements entails.

The result is a comprehensive approach in segmenting both the visual assets as well as the actions available on a user interface design into clearly defined, actionable and interpretable elements that are meaningful for the modelling procedure. We refer to sets of these elements as interface *states*.

### 3.2 User Modelling

With the interface segmentation defined as in Sect. 3.1, we are well prepared for a unified machine interpretation of a given design, thereby allowing for a full machine analysis of the described interface. This process has been initially developed by previous work [1]. Our contribution is predominantly in the area of interpreting the output of the previous stage and, more importantly, by introducing factors to the modelling procedure that allow for a representation of users with varying capabilities.

We accomplish this by utilising ACT-R [1], a Cognitive Architecture that has been in use (and actively developed) for decades and is particularly apt at giving insight to the simulated user behaviour of human-machine interfaces. The description to the modelling environment is accomplished by translating the XML-based descriptive factors into ACT-R-*interpretable* chunks that simulate both the input and output of a real user. Details of this process are outlined by previous work [11] and ample literature relating to ACT-R.

With the modelling process executed, the this stage of the framework outputs a fundamental dataset: the timings of each of the step of a user interface, given a specific route.

### 3.3 Usability Analysis

Routes and timings are particularly valuable in the process of assessing the usability of a given interface. Nonetheless, with these two sets of information segmented and hence assessed individually, their value is limited. The design specifications will provide all possible routes through the UI (the screen flow network). A user model will output, given a specific user interaction route, the timings associated with each step in the screen flow network. If this procedure is repeated many times, all possible routes a user may encounter given a specified interface, are covered by the model and hence a complete set of simulated user timings are captured.

This scenario of abundant, disjoint datasets raises the necessity for integrating the availability of scarce timings-data with the dense user journey data of the proposed interface design. More specifically, this means combining the timings captured from the modelling environment with the associated locations within a network of all available screens in the user interface. In previous work, we have called this process integrating the screen flow network with user data [12].

In the context of the framework introduced in this paper, we utilise an approach that is commonplace in mathematics and machine learning environments but not commonly used in the context of user interface development or usability analysis. By assigning a value to each stage (i.e. nodes) of the user interface that can be accessed by a user, a network is established. Next, all connections (i.e. vertices) between each of these states are analysed. This includes assigning a value that is based on the modelled user performance in the transition from each state to another. As defined in prior work [12], the analysis is performed in one of two modes: latitudinal progression or longitudinal progression. While the former represents the relative difficulty of a specific screen action for a specific user type, based on the overall results across multiple user types, the latter represents the probabilistic interpretation that a user will experience difficulties when transitioning from one screen to another.

With the network specified by the interconnectedness of the user interface and values to its vertices assigned by the two modes defined above, the values are propagated through the network utilising an adaptation of a message passing algorithm. This dynamically reassigns values for each of the nodes until the entire network of values converges (dependent on the quality of the modelled data). Once (and if) convergence is reached, the final node values represent a numerical classification of the likelihood that users will experience a bottleneck at each state of the user interface.

In summary, this generates one value associated with each screen or view (i.e. state) of the interface. This variable is representative of the relative complexity of that state compared to all other states (when assessed through longitudinal progression) or compared to a set of different user types (when assessed through latitudinal progression). We refer to sets of these descriptors as *complexity values*.

## 3.4 Design Injection

Complexity values can easily be made accessible through a dynamic, colour-coded or topological representation that indicates which screen requires the most attention at any point in the design process. While the focus of the development of the framework was not the design injection stage, we introduce a number of potential methods in which it can be integrated into the design process. This – in its conceptual design – is critical for the entire framework because it guides the overall flow of information and criteria for success of the associated research.

Two injection methods were analysed in the context of their feasibility and implementation potential: (i) a stand-alone representation on a dynamic, HTML5-based graph and (ii) the fully integrated approach, which is embedded in an integrated development environment (IDE) and dynamically outputs design recommendations as the interface is developed.

The stand-alone representation of the resultant, convergent values in a graph is useful if the analysis is performed once. The previous steps of the framework are entered manually by the designer or developer specifying the widgetised structure. The modelling and analysis phase are executed and a browser-based display shows a graphical network, based on the open source package Gephi. The visualisation contains nodes whose size are dependent on their interconnectedness, allowing the most frequently visited nodes to be analysed in the greatest depth. The colour of each node is dependent on the convergent complexity value (as defined in the previous Section). Here, it is important to highlight that the main intention of this approach is to provide an evidence-based visual summary of the components of a specified user interface design that are most difficult to reach, restricted by a set of assumptions set by the modelling environment. Both analysis modes, latitudinal progression or longitudinal progression can be displayed, allowing the analysis to favour either the progression according to an individual user's journey or the overall inclusivity of the design. Finally, based on user data, an age-corrective slider allows for the analysis to be adjusted for the anticipated age group of the modelled user.

The integrated, IDE-based integration of the framework serves a different purpose: it removes all friction from the modelling process, allowing the modelling to be executed frequently throughout the design process. The coded UI is extracted (in an automated fashion) from the development process and – once a property that solely connects screen actions with other screens – is established, translated into a simplified widgetised XML structure that can be interpreted by the modelling stage of the framework. Subsequently, the design is analysed and the results are displayed directly within the IDE. The representation of results is either in a list view that allows an ordering of screens based on their relative complexity values or in the same network representation that was introduced in the stand-alone variant described above. Additionally, a deep integration in established version control systems allows the tracking of UI design changes and the impact on complexity scores of individual screens. This allows for a design impact assessment throughout each iterative change of the design as well as a retrospective analysis thereof. Similarly to the stand-alone version of the design injection, an implementation of both usability analysis methods – latitudinal and longitudinal progression – can be modelled, as well as age-corrective measures for the intended age range of users.

## 4 Validation

The framework introduced in the previous section is partially validated through work relating to the individual stages and components of the framework. The approach of widgetising elements of a user interface through GOMS and various adaptions thereof is well established in literature. Similarly, the analysis of these widgetised structures has been implemented and validated in the context of follow-up work within the ACT-R modelling community, particularly through the implementation of the *device* module and its associated validation work [1]. Methods relating to message passing and, in particular, belief propagation, as implemented for the usability analysis of the work, are well established.

Because of the segmented nature of this prior validation work, a more complete validation of the pipeline was necessary in the context of the framework presented in this paper. In the context of the PhD Thesis on which this paper is based, a comprehensive user study was performed. Four distinct tasks on a mid-sized touchscreen device (1st Generation Apple iPad mini running iOS 7) were completed by a sample of users ($n = 28$) with a wide-ranging age distribution. This process had four distinct goals: (i) an analysis of the impact technology familiarity has on the performance of a task, (ii) an analysis of the impact age has on the performance of a task, (iii) the validation of the ACT-R modelling implementation introduced in this paper, and (iv) the validation of the usability analysis sub-stage introduced in this paper. While the specifics of this comprehensive study are beyond the scope of this paper, we present a summary of the methods used, as well as the key results.

### 4.1 Experimental Procedure

The participants were, following a brief outline of the work and written consent, asked to complete a technology familiarity questionnaire (TFQ), adapted from the version introduced by Blackler [2]. This was completed on a computer and included questions that related to the self-perceived competence as well as frequency of use of a number of devices and device classes, ranging from mobile phone to desktop computer. The results were scored using the same criteria as in previous work using this method of assessing technology familiarity [2].

Subsequently and in a randomised order to minimise ordering effects, each participant completed four tasks. Task one required the user to set an alarm using the built-in operating system's alarm functionality. Task two made use of the operating system's built-in reminders functionality and required the user to create three reminders and mark two as completed. Task three asked the user to find, order, and process an item in *Amazon*'s shopping application. The fourth and final task was split into three sub-tasks within the *Bloomberg* application. This involved looking up the value of a specific bond, reading the news headlines, and adding a stock to the built-in watch-list functionality of the app.

For each of the tasks, the user was observed using a wireless screen capture of the device, a video capture from above the device, and an audio recording. This allowed both the timings and route choices through each of the tasks to be captured and analysed. Finally, after the participant completed each of the tasks, a retrospective video analysis was performed. By presenting the user with a recording of his or her own interaction with each of the tasks (in the order the tasks were completed), the participant could comment on each of the steps previously undertaken. In addition to the quantitative data that resulted from the timings and route-choice analysis, this qualitative dataset allowed the precise location their usage bottleneck to be communicated, without the limitation of having to infer the location of greatest perceived complexity based on performance timings.

## 4.2 Results

While the learnings from the experimental work associated with the framework are wide-ranging, we focus on four key results: (i) the impact the TFQ had on the performance of each task, (ii) the impact the age of the participant had on the performance of each task, (iii) the validity of the timings that were generated by the model, based on the captured user timings and (iv) the validity of the complexity scores that were generated by the model, based on the captured user locations of perceived complexity through the retrospective video analysis stage of the experimental work.

*Result 1: TFQ and User Performance* Using the TFQ score of each participant and each participant's performance scores, based on the timing distribution of all steps of each completed task, we analysed the relationship between user performance and technological familiarity. The TFQ score was based on an adapted version of previous work by Blackler [2], as well as sub-scores thereof. We concluded that there is no significant effect of TFQ on user timings across all four tasks.

*Result 2: Age and User Performance* There was a significant difference in performance timings across three of four tasks based on three age levels ($p = 0.014$). More specifically, a correlative analysis thereof allowed us to specify a linear relationship between these two variables to some degree of certainty ($r^2 = 0.48$).

*Result 3: ACT-R Model Validation* Mean modelled performance timings from the ACT-R model correlated in trend with mean user performance timings of all users. Despite a comparably low correlative value ($r^2 = 0.61$), the ordering of empirical user performance and modelled user performance matched in three out of four tasks, across all participants.

*Result 4: Usability Analysis Validation* The most frequently self-defined locations of greatest complexity by participants for each task, matched three out of four tasks' usability analyses. More specifically, for two out of four tasks, the distribution of self-defined locations of greatest complexity was indicative of the modelled distribution of complexity values.

*Other Results.* Furthermore, insights on repeated actions within one task were acquired. These include that repeated tasks show increasing performance gains within one session, independent of whether the user had experience with the specific interface or screen network before. Further research is required to translate these validated hypotheses into concrete, validated numerical implementations of the model.

### 4.3 Discussion

The framework introduced in this paper is novel in its composition and application whilst being built utilising well-established methods. Hence, the holistic validation of the framework, as outlined in this Section is essential for its application. The clarity of Results 3 and 4 speak for themselves in terms of the applicability of the overall approach in the context of informing and improving user interface designs. It is clear that the timings – corrected for age – generated by the model are suitably indicative for the model to be utilised for the purpose defined in this paper. Similarly, the usability analysis performed in the latter stage of the framework produces a prioritisation of the four tasks (and their subtasks) that is suitable for locating potential performance bottlenecks once utilised by real users. It is important to note in this context that the results are constrained by the small sample size as well as by the specificity of the tasks chosen to test the framework on. Some of the tasks had been used by many of the participants before. Despite this, the overall trends across tasks remained constant, with one notable exception of task four. This is important because it underlines the independence of prior use on the model's validity, albeit only to the level of accuracy the model can provide.

The difference between the two modelling approaches – latitudinal and longitudinal progression – could not be established given the design of the experiment. While longitudinal progression focusses on a specific user type, latitudinal progression focusses on a specific subsection of the screen network. Given the sample size, these two factors could not be clearly segmented in the performance data and hence no conclusive findings could be made in relation to the two approaches. However, their combined validity was established. Further work is required to assess each individually.

Two additional insights generated by the validation stage included the impact of technological familiarity and age of the participants had on their performance on each of the tasks. It is clear that across all tasks, there was a monotonic relationship between age and performance timing. Additionally, we were able to define an indicative value of this relationship across three of four of the tasks for all users with a high correlative value, allowing an "age coefficient" to be defined. This means that the modelling output can be corrected for the age of the user through a linear transformation.

In contrast, no clear relationship between the TFQ and user performance was found. This may be a result of the method (the TFQ structure and/or scoring) or indicative of that technological familiarity has little to no impact on the performance of the tasks tested. To further this analysis, multiple scoring methods and sub-scores of the TFQ were tested in the context of performance data, resulting in the same result.

## 5 Conclusion

In this paper we introduce a framework that can be integrated into existing design processes of user interface designers and developers to improve the design of a given interface. We outline the DRM to be the underlying methodology and introduce the four stages of the model. Finally, we outline the process and key results of a validation study in this context.

We determine that while there is no clearly identifiable relationship between technological familiarity and task performance, there is a clear trend between the age of the user and the performance timings. Additionally, we identify that with these factors applied, a model of the user employing ACT-R can be utilised to indicate trends in timings of real users. By numerically combining the ACT-R model with a graphical representation of the modelled interface, the hotspots of the design can be indicatively identified. Finally, two potential methods of integrating the resultant performance values into the design process of interfaces are outlined, but remain to be fully implemented and validated.

Whilst the work, and in particular the validation work, introduced in this paper concentrates on the validation of the underlying processes (such as specific models and mathematical methods), further work will focus on a validation of the complete pipeline employed in design processes and hence provide the final necessary step to translate these findings into practice. Further work will also focus on extending the validation, such as increasing the number of data points (i.e. number of participants), introducing additional validation tasks with lower degrees of specificity and extending the validation procedure to the design impact of the entire framework.

## References

1. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychol. Rev. **111**, 1036–1060 (2004)
2. Blackler, A.: Intuitive interaction with complex artefacts. Ph.D. thesis. Queensland University of Technology, Australia (2006)
3. Bracewell, R., Shea, K., Langdon, P., Blessing, L., Clarkson, P.: A methodology for computational design tool research. In: 13th International Conference on Engineering Design (ICED 2001), Glasgow, Scotland, UK, pp. 181–188 (2001)
4. Card, S.K., Moran, T.P., Newell, A.: The Psychology of Human-Computer Interaction. Erlbaum (1983). http://books.google.at/books?id=qU-DaL49R9EC
5. Card, S.K., Moran, T.P., Newell, A.: The keystroke-level model for user performance time with interactive systems. Commun. ACM **23**(7), 396–410 (1980). http://doi.acm.org/10.1145/358886.358895
6. John, B.E.: CogTool User Guide. Technical report (2009)

7. John, B.E., Gray, W.D.: CPM-GOMS: an analysis method for tasks with parallel activities. In: Conference Companion on Human Factors in Computing Systems. CHI 1995, pp. 393–394. ACM, New York (1995). http://doi.acm.org/10.1145/223355.223738
8. Kieras, D.: A guide to GOMS model usability evaluation using NGOMSL. In: Helander, M.G., Landauer, T.K., Pradhu, P.V. (eds.) Handbook of Human-Computer Interaction (2nd ed.), pp. 733–766. North-Holland, Amsterdam (1997)
9. Schrepp, M.: GOMS analysis as a tool to investigate the usability of web units for disabled users. Univ. Access Inf. Soc. **9**(1), 77–86 (2009)
10. Wollner, P.K., Goldhaber, T., Mieczakowski, A., Langdon, P.M., Hosking, I.M., Clarkson, P.J., et al.: Evaluation of setup procedures on mobile devices based on users initial experience. In: The 9th NordDesign Conference on DS 71: Proceedings of NordDesign 2012, Aarlborg University, Denmark, 22–24.08.2012 (2012)
11. Wollner, P.K.A., Hosking, I., Langdon, P.M., Clarkson, P.J.: Improvements in interface design through implicit modeling. In: Stephanidis, C., Antona, M. (eds.) UAHCI/HCII 2013, Part I. LNCS, vol. 8009, pp. 127–136. Springer, Heidelberg (2013)
12. Wollner, P.K.A., Langdon, P.M., Clarkson, P.J.: A combinatory approach to assessing user performance of digital interfaces. In: Langdon, P.M., Lazar, J., Heylighen, A., Dong, H. (eds.) Inclusive Designing, Part II, pp. 39–48. Springer International Publishing, Switzerland (2014)