A Combinatory Approach to Assessing User Performance of Digital Interfaces

P. K. A. Wollner, P. M. Langdon and P. J. Clarkson

Abstract Digital devices are often restricted by the complexity of their user interface (UI) design. While accessibility guidelines exist that reduce the barriers to access information and communications technology (ICT), guidelines alone do not guarantee a fully inclusive design. In the past, iterative design processes using representative user groups to test prototypes were the standard methods for increasing the inclusivity of a given design, but cognitive modelling (the modelling of human behaviour, in this instance when interacting with a device) has recently become a feasible alternative to rigorous user testing (John and Suzuki 2009). Nonetheless, many models are limited to an output that communicates little more than the assumed time the modelled user would require to complete the task given a specific way of doing so (John 2011). This chapter introduces a novel approach that makes use of the overlay of user modelling output (timings) onto a graphical representation of an entire UI, thereby enabling the computation of new metrics that indicate the relative inclusiveness of individual screens of the UI.

1 Introduction

Digital devices are often restricted by the complexity of their user interface (UI) design. While accessibility guidelines exist that reduce the barriers to access information and communications technology (ICT), guidelines alone do not guarantee a fully inclusive design. In the past, iterative design processes using representative user groups to test prototypes were the standard methods for increasing the inclusivity of a given design, but cognitive modelling (the modelling of human behaviour, in this instance when interacting with a device) has

P. K. A. Wollner (✉) · P. M. Langdon · P. J. Clarkson
Engineering Design Centre, Department of Engineering, University of Cambridge, Cambridge, UK
e-mail: pkaw2@cam.ac.uk

P. M. Langdon et al. (eds.), *Inclusive Designing*, DOI: 10.1007/978-3-319-05095-9_4,

recently become a feasible alternative to rigorous user testing (John and Suzuki 2009). Nonetheless, many models are limited to an output that communicates little more than the assumed time the modelled user would require to complete the task given a specific way of doing so (John 2011).

In this chapter, we present a novel way of combining the output of user simulations into a unified mathematical representation, allowing for a more holistic interpretation of the simulation results. From a universal access perspective, this contributes to the field through improved accessibility of product designs which employ this novel approach. This is achieved through a more complete understanding of models that are based on specific impairments, as postulated by Langdon and Thimbleby (2010) and further explored by work such as Biswas et al. (2012). While we present this method to be utilised in combination with cognitive architectures (toolkits that combine theories of cognition to simulate human behaviour), it also may be presented as a novel way of interpreting studies involving real users and their performance data.

This chapter outlines (i) the structure and function of cognitive architectures, (ii) how these architectures can be employed to simulate user behaviour whilst interacting with touchscreen devices, (iii) the limitations of the output data (particularly from a universal access perspective), (iv) a proposed extension based on the application of graph theory on the output data and (v) how this mathematical model can be communicated to designers and developers in order to inform universally accessible designs.

2 Cognitive Architectures

Cognitive architectures are toolkits that combine theories of cognition to simulate human behaviour. The simulations are not purely a model of the behavioural output of the human mind but rather aim to replicate the structural properties of the modelled system. They do not physically replicate the components of the system—rather, virtual machines replicate the behaviour and knowledge of humans. By employing these frameworks to 'experience' the stimuli of a specific human–machine interaction scenario and in turn 'act' upon these stimuli, valuable user performance data can be generated by the model. This data is often represented as timings, based on a predefined set of actions which the architecture simulates, evaluates and outputs as a time value, indicating how long a real user would require to complete the same task.

Employing cognitive architectures in the design of interactive products and services allows the designer to gain insight on user behaviour without requiring the physical presence of users. Using a modelling approach as a replacement for user testing provides multiple benefits for both designers and—through the creation of improved products—for users.

Benefits for the designer include a process that supports quicker design iterations through the reduction of time-consuming participant recruitment and testing.

Furthermore, due to the ability to extend the model to factor in elements of the simulated users' prior knowledge, the resulting performance data can be tailored to the specific abilities of a subset of real users. This supports the design of interfaces targeted specifically at particular impairments, aiming to fulfil the overarching goal of more inclusive designs.

Benefits for the user may be defined in a similar way; the product better meets the cognitive requirements of a subsample of users based both on the cognitive impairments the users may have and the impact prior experience (Langdon et al. 2010) may have on their ability to interact with a new interface design. Benefits for the user, once the design is completed, include an optimised experience that is designed with the specific user type in mind.

While there are a number of cognitive architectures that are suitable for the automated performance evaluation of UI designs, we base much of the subsequent discussion on the assumed utilisation of ACT-R, one of the most extensive and evolved cognitive architectures (Anderson et al. 2004). This is based on the environment in which the related testing is performed (see Wollner et al. 2013) and represents the framework in which the model introduced in this chapter will be tested and deployed.

Originally, more advanced modelling tools such as ACT-R (which includes capabilities for simulated knowledge acquisition, i.e. the ability to learn and act upon learned facts) required extensive experience with the modelling language and the underlying assumptions in order to simulate cognitive processes accurately.

This implies that the entirety of the interaction process and environment are translated into a form that the architecture can interpret. Hence, this requires experts in these architectures to manually perform the translation and makes the potentially valuable data output inaccessible to most designers and/or developers.

Recently, cognitive models have become more accessible to designers with limited or no modelling experience (Councill et al. 2003); examples of this include Salvucci and Lee's (2003) ACT-Simple which uses a KLM-GOMS-based (John et al. 2004) descriptive language to automatically translate a specific UI design into a form that can be utilised as the basis of the simulation environment. Despite this simplification, there is still an inaccessibility in the assessment and communication of the output of cognitive models. Large amounts of segmented user data, mainly based on individual timings of actions within the UI, are presented to the designer without further analysis.

2.1 Screen Flow Network

The basis for this chapter is extending and combining the output of cognitive architectures. This is provided by establishing a graphic representation of the interaction paradigms and interaction routes available on digital interfaces. We present two types of elements included in this abstraction of UI progression: (i) elements that represent individual screens and (ii) connections that represent

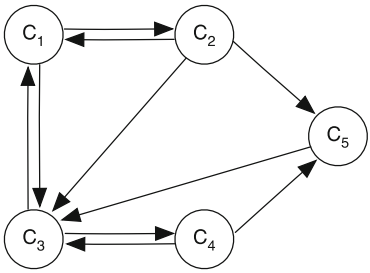

Fig. 1 In this screen flow, network circles represent individual screens within a UI progression, the lines represent the directionality of actions on individual screens that allow movement to another. E.g. C_1 has buttons or provides an implicit function to move to C_2 and C_3; C_5, in turn, only allows movement to C_3

actions that the user may complete when moving between these screens, including data input such as keyboard entry or drawing on the screen. The preliminary work (before running a cognitive model) builds this representation, the subsequent simulation extends the aforementioned connections between UI screens with values that represent the time a simulated user may require to move between screens. We refer to this as a 'screen flow network', as depicted in Fig. 1.

Despite the fact that the network displayed in Fig. 1 is not fully representative of a UI (in lacking the option to include global actions, such as a 'home' button and/or virtual global actions, such as an on-screen keyboard), it enables the representation of most actions that are based on the transition from one UI screen to the next. Further work will gradually extend the screen flow network to allow the representation of global actions.

Through the network introduced above, the designer can, through visual methods, explore design alternatives based on hot spots within the UI progression (indicated by timings). While this improves the design process, it cannot represent the entire network of possible screen flows the user may choose to explore. This is because the cognitive modelling approach is limited to a specific screen progression that the simulated user navigates through rather than making all screen flows possible. This means that the screen flow network provides a visual model indicating all possible screen flows given a specific UI design, but the modelled output of the same UI design is limited by the restricted information cognitive architectures can provide.

3 Simulation Network Analysis

Given the mismatch of the available and exploited data presented in the previous section, we propose to use a graph theoretical approach to make use of the overlay of user modelling output (timings) onto the entire screen flow network. This is

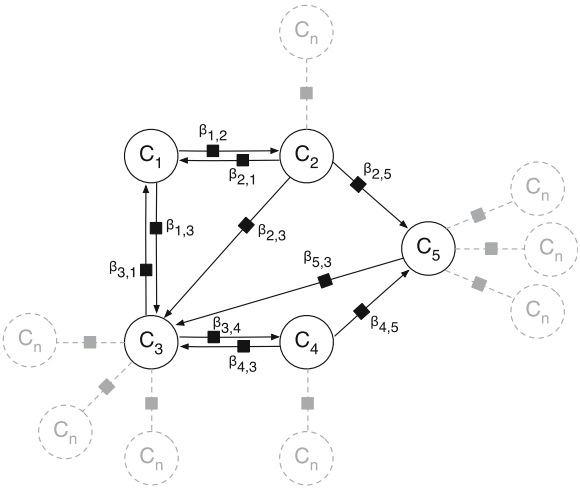

Fig. 2 Intersection network used in factor graph notation, where the arrows indicate actions that allow the navigation between screens (denoted by C_n) and the variables β_{n_1,n_2} are defined by the simulation output

possible through the mapping of individual screens and their connections. We propose that the user modelling is executed on all screen progressions (rather than one specified route) and the resultant progression timings are included as weightings of the connecting elements of the screen flow network.

More specifically, we propose to integrate the availability of rich data regarding the interconnectedness of individual screens and the sparse timings-based data that is available as an output of user modelling of interfaces. Vertices in this representation match individual screens within the context of the application; edges are the actions the user may complete to move between these screens. The edges are reweighted by each iteration of implicit modelling (user simulations), allowing—once a critical mass of simulations has been reached—a representation of both user progression and potential (timing-based) performance bottlenecks within.

Figure 2 outlines the factor graph approach in more detail. The vertices C_n represent individual screens within the UI flow. Screens are connected by edges that have a weighting β_{n_1,n_2}, which represents the relative complexity of moving from vertex C_{n_1} to C_{n_2}. The diagram indicates only one-directional weightings, in cases, where there are no actions available to move to another screen. In the example shown in Fig. 2, this applies to screen C_5, which can, for instance, only be reached from screens C_2 and C_4. Hence the weightings $\beta_{2,5}$ and $\beta_{4,5}$ are non-zero. The weightings $\beta_{5,2}$ and $\beta_{5,4}$ are zero and hence not depicted in the diagrammatic representation. In contrast, screen C_5 only provides a direct action to move to screen C_5, which, in turn, is indicated by the non-zero weighting $\beta_{5,3}$.

In this context, we need to define the factor graph theoretical relationships that govern the weightings introduced in Fig. 2.

3.1 Longitudinal User Progression (β_{n_1,n_2})

The probabilistic interpretation that a user will experience difficulties when transitioning from screen C_{n_1} to screen C_{n_2} may be represented by the longitudinal user progression introduced in this section. By determining the timing of an individual screen change within the screen flow network, this descriptor can be numerically assessed based on the distribution of timings across all actions in the network. The descriptor is outlined in Eq. (1) where S is the set of all possible timed actions (edges) within the screen flow network and t_{n_1,n_2} is the timing for transitioning from vertex C_{n_1} to C_{n_2}.

$$\beta_{n_1,n_2} = \left[\frac{\sum_{i \in S} t_i}{|S|\left(t_{n_1,n_2}\right)} \right] \tag{1}$$

In other words, the descriptor introduced compares the simulated time an individual action takes in relation to the mean time of all available actions in that screen flow. This descriptor does not account for variations in user type.

3.2 Latitudinal User Progression ($\beta_{n_1,n_2}(u)$)

The probabilistic interpretation of the relative difficulty of a specific screen action for a specific user type u, based on the overall results is defined as latitudinal user progression. This is determined by comparing the variation in user timings based on the non-linear difference of timings of all user types, adjusted linearly by $\alpha(u)$, and compared to the mean timing of that action.

$$\beta_{n_1,n_2}(u) = \left[\frac{\sum_{v \in U} t_{n_1,n_2}(v)}{\alpha(u)|U|\left(t_{n_1,n_2}(u)\right)} \right] \tag{2}$$

In this representation, $\alpha(u)$ (defined in Eq. 3) is a function to correct for the specific user type, U is the set of all tested user types and $t_{n_1,n_2}(u)$ is the timing for transitioning from vertex C_{n_1} to C_{n_2} for user u.

$$\alpha(u) = \left[\frac{\sum_{v \in U} \sum_{i \in S} t_i(v)}{|U| \sum_{l \in S} t_l(u)} \right] \tag{3}$$

S is the set of all possible timed actions (edges) within the screen flow network and t_i is the timing for transitioning within vertex i.

In other words, the descriptor introduced in Eq. (2) is determined by the simulated time required for one action to be completed by one user type compared to the mean time of the same action for all other user types. This mean time is corrected to account for the overall (non-task specific) performance differences between user types by $\alpha(u)$ (defined in Eq. 3).

3.3 Comparison of Models

For both models, we have a simple distribution of values that allows the assessment of individual stages within the screen progression:

$\beta_{n_1,n_2} = 0$	It is impossible to progress from vertex C_{n_1} to C_{n_2}
$0 < \beta_{n_1,n_2} < 1$	It is difficult to transition from vertex C_{n_1} to C_{n_2}
$\beta_{n_1,n_2} = 1$	It is possible to transition from vertex C_{n_1} to C_{n_2}
$1 < \beta_{n_1,n_2}$	It is easy to transition from vertex C_{n_1} to C_{n_2}

Furthermore, there are subtle differences between the two models introduced in the sections above; while the first model is defined for only one user type, it is limited to compare performance timing only to all other actions within the same screen flow network.

The second model utilises a descriptor that compares the performance of a particular action within the screen flow network across all user types that were simulated.

While the first model highlights individual elements of the screen flow network that will—in relative terms—be more complex to a user and hence improves the usability across an interaction session, the second model assesses the complexity across user types and is more relevant for ensuring an inclusive design of the simulated interface.

Both models are constrained by utilising solely the mean timings (of all actions for one user type and for one action across all user types, respectively) rather than integrating the distribution of timings of these measures.

3.4 Complexity Propagation

Using the above-defined complexity models, we may define a formulaic approach for assigning complexity values to the vertices C_n based on the weightings (β) of the edges between them. For this, we resort to a message passing algorithm. If we assume the overall network presented in this chapter to be a bipartite graph where all nodes are either screens, denoted by the set C, or connections between screens (β), we may use the concept of belief propagation to evaluate the values of all screens. We call this concept complexity propagation. Here, to align the notation with that of Pearl (1982), we propose to denote screens as factors U and the movement between screens as variables V.

First, it is necessary to define a joint mass function, which is determined by the weighting of connections between screens, given a current valuation of the screen node they are connected to.

Next, the entire network is evaluated, by setting the values of each screen node, C_n, to a initialisation value and a message passing algorithm is employed until convergent values of the screen nodes are reached. This means that the screen nodes are revalued based on messages based through all neighbouring factor nodes β_{n_1,n_2}. These complexity messages are denoted as $\mu_{C\to\beta}$ and $\mu_{\beta\to C}$, respectively. The governing equations for this process are defined by Pearl (1982) (and subsequent work) and extend beyond the scope of this chapter.

4 Design Process Integration

Using the resultant graphical model permits the exploitation of various graph theoretical methods (Bondy and Murty 2008) that allow not only the definition of optimal routes of UI progression but also the topographical mapping of UI complexity, a method that provides a simplistic, data-rich representation of user complexity through screen flow networks. Three-dimensional mapping adds another dimension to the screen flow representation in edge and vertex form. The same concept as a standard network is utilised, but the third dimension is qualified by the distribution of weightings of all connected edges. Hence, the most altitudinal points on the visual representation relate to vertices (or screens) that are the most difficult to reach.

This representation provides a model for UI designers to better understand the limitations of a proposed design (given the simulation output thereof) without abstracting the key output of user modelling: timing-based data. While previous work (such as Thimbleby 2010) has suggested a purely graph theoretical approach to UI assessment, we propose a direct interpretation based on the timings-based output of cognitive models.

Here, we can take the final values of factor nodes (the values of screens C_n after the previously introduced complexity passing algorithm converges) to indicate the assumed complexity of a specific screen flow. This means that the relative complexity value established may be presented visually to the designer, allowing him or her to better comprehend the output of the simulation, translation and complexity passing procedure introduced in this chapter.

This could be visualised in a simple three-dimensional framework, where the x- and y-coordinates are determined arbitrarily (similarly to the screen flow network introduced earlier in this chapter) and the z-coordinate is established by a normalised representation of the factor value, post-convergence. A simplified example of such a representation is depicted in Fig. 3.

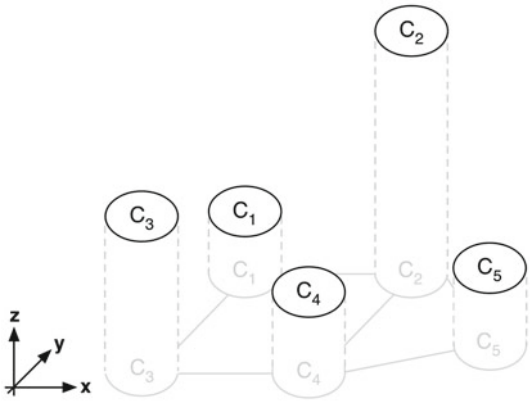

Fig. 3 Three-dimensional complexity representation of the screen flow network introduced in Fig. 1, where the height (along the *z-axis*) defines the complexity of reaching the given screen. In this example, screen C_2 is the one that is most difficult to reach

5 Conclusion

This chapter presents the mathematical foundation of a graph theoretical approach for assessing UI complexity, given the timings-based output of cognitive simulations. It outlines (i) the framework in which a screen flow network is constructed, (ii) the method by which individual cognitive simulations are instantiated and how algorithmically all possible flows are processed, (iii) the approach by which the resultant user progression data (i.e. timings) may be agglomerated into weightings, (iv) the way in which probability of user access, β_{n_1,n_2}, may be identified and (v) recommendations for further graph theoretical operations, given the completed network.

We believe that this novel approach supports the design process of inclusive user interfaces through two key mechanisms: representation and adaptability. Representation is the process in which segmented numerical output data (i.e. timings from cognitive architectures) is collated in a visual representation which supports the designer in making better decisions based on data that gives insights to a broad range of users. Adaptability relates to the underlying strength of cognitive architectures, which allow specific impairments to be modelled, giving the designer the ability to investigate the usability of an interface for specific user types.

Further work will focus on the extension of the complexity passing algorithm introduced in this chapter and in developing a standardised three-dimensional representation similar to Fig. 3 in order to communicate the output data of this method to designers. Additionally, the computational framework may be extended to be utilised not only with the output of cognitive architectures but also with user performance data gained by actual user tests. This would require an adaption of the user type variable introduced earlier in this chapter, but would greatly extend the applicability of user testing in the design process.

References

Anderson JR, Bothell D, Byrne MD, Douglass S, Lebiere C et al (2004) An integrated theory of the mind. Psychol Rev 111:1036–1060

Biswas P, Robinson P, Langdon PM (2012) Designing inclusive interfaces through user modeling and simulation. Int J Human-Comput Interact 28(1):1–33

Bondy A, Murty U (2008) Graph theory. In: Graduate texts in mathematics. Springer, New York

Councill IG, Haynes SR, Ritter FE (2003) Explaining soar: analysis of existing tools and user information requirements. In: Proceedings of the 5th international conference on cognitive modeling

John BE (2011) Using predictive human performance models to inspire and support UI design recommendations. In: Proceedings of the ACM CHI conference on human factors in computing systems, Vancouver, Canada

John BE, Prevas K, Salvucci DD, Koedinger K (2004) Predictive human performance modeling made easy. In: Proceedings of the SIGCHI conference on human factors in computing systems, New York, NY, US

John BE, Suzuki S (2009) Toward cognitive modeling for predicting usability. In: Human-computer interaction. New trends. Lecture notes in computer science, vol 5610, pp 267–276

Langdon PM, Persad U, Clarkson PJ (2010) Developing a model of cognitive interaction for analytical inclusive design evaluation. Interact Comput 22(6):510–529

Langdon PM, Thimbleby H (2010) Inclusion and interaction: designing interaction for inclusive populations. Interact Comput 22(6):439–448

Pearl J (1982) Reverend bayes on inference engines: a distributed hierarchical approach. In: Proceedings of the American association of artificial intelligence national conference on AI, Pittsburgh, PA, US

Salvucci DD, Lee FJ (2003) Simple cognitive modeling in a complex cognitive architecture. In: Proceedings of the ACM CHI 2003 human factors in computing systems conference, Ft Lauderdale, FL, US

Thimbleby H (2010) Press on - Principles of interaction programming. MIT Press, Cambridge

Wollner PKA, Hosking I, Langdon PM, Clarkson PJ (2013) Improvements in interface design through implicit modeling. In: Universal access in human-computer interaction. Design methods, tools, and interaction techniques for eInclusion. Lecture notes in computer science,vol 8009, pp 127–136