Available online at www.sciencedirect.com
SciVerse ScienceDirect
Int. J. Human-Computer Studies 71 (2013) 276–302
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

Action graphs and user performance analysis

Harold Thimbleby
Future Interaction Technology Laboratory, College of Science, Swansea University, Swansea, Wales SA2 8PP, UK

Received 23 December 2011; received in revised form 24 September 2012; accepted 12 October 2012
Communicated by S. Wiedenbeck
Available online 7 November 2012

Abstract

A user operating an interactive system performs actions such as "pressing a button" and these actions cause state transitions in the system. However to perform an action, a user has to do what amounts to a state transition themselves, from the state of having completed the previous action to the state of starting to perform the next action; this user transition is out of step with the system's transition. This paper introduces action graphs, an elegant way of making user transitions explicit in the arcs of a graph derived from the system specification. Essentially, a conventional transition system has arcs labeled in the form "user performs action *A*" whereas an action graph has arcs labelled in the form "having performed action *P*, the user performs *Q*." Action graphs support many modelling techniques (such as GOMS, KLM or shortest paths) that could have been applied to the user's actions or to the system graph, but because it combines both, the modelling techniques can be used more powerfully.

Action graphs can be used to directly apply user performance metrics and hence perform formal evaluations of interactive systems. The Fitts Law is one of the simplest and most robust of such user modelling techniques, and is used as an illustration of the value of action graphs in this paper. Action graphs can help analyze particular tasks, any sample of tasks, or all possible tasks a device supports—which would be impractical for empirical evaluations. This is an important result for analyzing safety critical interactive systems, where it is important to cover all possible tasks in testing even when doing so is not feasible using human participants because of the complexity of the system.

An algorithm is presented for the construction of action graphs. Action graphs are then used to study devices (a consumer device, a digital multimeter, an infusion pump) and results suggest that: optimal time is correlated with keystroke count, and that keyboard layout has little impact on optimal times. Many other applications of action graphs are suggested.

Keywords: Action graph; The Fitts Law; Finite state transition system; Lower bounds on task time; User modelling

1. Introduction

Predicting how users will perform using an interactive system is a key part of the science of HCI as well as a practical part of usability analysis.

This paper introduces action graphs, which generalise finite state machines to allow analysis of user actions. The dimensions of buttons and their separation along with an action graph can be used to predict time or other costs the user incurs for any sequence of activities. Since times are calculated using programs, *any* programmable function can be used, such as the Fitts Law, KLM or other model (even financial costs). This paper provides an algorithm (in Java) to convert a standard model into an action graph; our work is reproducible and could be embedded into analysis tools. This makes a significant advance on our previous work (Thimbleby, 2007a; Gimblett and Thimbleby, 2010).

Almost any interactive system can be analyzed with action graphs, though the example case studies in this paper are based on "control panel" devices with a small keypad, rather than typewriter (QWERTY)-based devices; thus this paper is not explicitly concerned with information-based applications (word processing, data entry, diaries, address books, etc.) but with the control of systems (such as instrumentation, medical devices, consumer devices)—although an abstract view of a complex application such as a word processor may have interesting control features,

E-mail address: harold@thimbleby.net
URL: http://harold.thimbleby.net

1071-5819/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ijhcs.2012.10.014

say, in its dialog or menu structures, which would be amenable to action graph analysis.

Our action graph case studies suggest that optimal task time and keystroke counts are correlated and, surprisingly, that keyboard layout is not a significant factor in optimal task times. However, such results are but a small contribution of the paper, since action graphs can be used to explore many further issues.

1.1. Conceptual background: tasks, generalisations and abstractions

This paper presents a mathematical framework to address certain HCI questions, and its main benefits are that it permits a complete and automatic analysis of certain issues previously beyond the reach of researchers (except in the very simplest of cases). As a piece of mathematics, it is correct; the key questions, then, are whether it can be applied to HCI in an appropriate and in a useful way?

By way of comparison, ‘‘addition’’ is correct mathematically, but whether and to what extent it can be usefully applied to real-world questions, say, about money and cash depends on various non-mathematical, or at least ‘‘non-addition’’ issues. For example, in ‘‘the real world’’ there are inflation and interest and bank charges and even thieves, so money in a bank account does not quite follow the usual laws of addition without a lot of qualification. Cash, however, is more familiar than HCI theory and certainly far clearer than action graphs, which this paper introduces! We will therefore use the very familiar territory of cash as a conceptual bridge to help make some of the HCI issues of action graphs clearer: very familiar issues with cash and addition have interestingly analogous issues in the less familiar territory of action graphs.

The big picture could be put like this: although one would hardly think of dismissing the abstract idea of addition because of the technicalities of inflation, it might be tempting to dismiss action graphs because of ‘‘their’’ problems when in fact the problems are more to do with the complexity of HCI. In particular, the rigour of action graphs highlight many boundary problems that deserve more research, in a sort of similar way that an apparent failure of addition on your bank account might reveal a thief or something even more interesting at work that deserves closer investigation rather than dismissing theory that does not cover everything.

If different sorts of coins are to be added for a cash value, they should be treated with different values. In this paper, we use our approach to add times due to finger movement, but it could also be used to add times (or even cash values) from other sources. We use the Fitts Law to estimate times for a user to do tasks, but we could have used, for instance, KLM (Card et al., 1980), which would add further types of time values. Mathematically, this is trivial, but for the first paper introducing the approach it adds a level of complexity—in fact, we side-step this complexity by emphasising lower bounds on times; KLM would *increase* timing estimates, but does not affect hard results from lower bounds. (The second case study, discussed in B.1, introduces ‘‘button hold operators,’’ which shows that generalizations like KLM are trivial.)

If cash (e.g., from a loan) is to be added, it may have a time-dependent value. We assume the user interface has a fixed physical layout, as occurs on physical devices such as industrial control panels. The mathematics can handle dynamic, soft key layouts, too, but for the purposes of this paper such dynamic features introduce unnecessary complexity.

If very large amounts of cash are to be added, a computer program may overflow and give incorrect results. We use a computer to perform calculations with action graphs, and as such, we are limited to work within the practical limitations of computers. This means there are some interactive systems that are too complex to be satisfactorily analyzed, but we would argue that such systems raise HCI questions of a different nature than our approach is intended to handle. Moreover, a system that is too complex for a computer to analyze is possibly too complicated for conventional concepts of usability to be applied.

If people do not declare all their capital and cash flow, one will obtain incorrect results. People often ignore illiquid capital—because they are only interested in cash, or perhaps because they are trying to pay smaller insurance premiums. In other words, one has to be clear what the task is, and then analyze it correctly. Our approach uses action graphs. Any task a user performs on a device changes the state of the device's action graph; thus, every task corresponds to a state change. Just as there are some types of monetary value one may not wish to declare, there are some types of state change that one may—or may not—consider to be valid tasks. For example, a type of task one might want to ignore when analysing a ticket machine is ‘‘press buttons, insert cash, but do not get a ticket.’’ Undoubtedly the device has a sequence of states corresponding to this failed task! For some analyzes, one might want to know the time cost to the user of failure (presumably it would be very frustrating for it to take a long time before the user discovers they cannot get a ticket), and for other analyses one might wish to ignore it. From a computational perspective, both choices are easy: we can define tasks as any state change, or define tasks as any state change ending with dispensing a ticket—or we can impose any other task criterion that interests us to analyse.

How people wish to use their cash is a question of economics, not just of addition. What tasks a user wishes to perform is beyond the scope of this paper.

Some people may be quite happy not knowing exactly how much cash they have; they do not need to use addition (adding up coins), they just shake and listen to the piggy bank, or use some other heuristic to check they have enough to live by. Although action graphs give precise answers to certain HCI questions, indeed questions that previously