mind the gap – exploring the void of the task counter

Position Paper – HCI2003 Workshop on the Temporal Aspects of Tasks

Alan Dix

Lancaster University, UK
http://www.hcibook.com/alan/topics/time/

1. Introduction

Issues concerning time in the user interface have been a recurrent theme in my research ever since my "Myth of the Infinitely Fast Machine" paper in 1987 [3]. In this paper I'm going to focus on those aspects concerned with the precise timing of tasks and in particular at a close examination of the gaps between tasks.

In work over several years with Devina Rambuny-Ellis and Julie Wilkinson, we have looked at the nature of this gap and in particular what makes tasks happen when they happen. We call this *Trigger Analysis* and a chapter describing it will appear in the new Diaper and Stanton task analysis collection [12,2]. In more recent work with Cristina Chisalita and Gerrit van der Veer, we have looked at the relationship between events and triggers in Trigger Analysis and similarly named concepts in Groupware Task Analysis [20,21]. In grappling with the similarities and differences between these we have been led to a more rich understanding of events and the way events drive and are driven by tasks. One outcome of this is a more detailed analysis of the gap between tasks and a resulting view of the life history of a task [11].

2. Background

Before focusing on the gaps between tasks, I will give a brief overview of some of the other aspects of interface timing in my own work that are related to or lead into the analyses that are the heart of this paper.

2.1 pace and tasks

In "Pace and Interaction" in 1992 I investigated the issues concerning the pace of communication channels and how this affects cooperative tasks [5]. The pace of a channel is the rate at which one can expect a round trip message. that is for a message sent to be received, understood acted upon and the feedback to be received. This depends on a number of factors such as the delays in the underlying communication infrastructure (e.g. post needs at least two days for round trip) but also including the interface (e.g. if new email generates an alert it may be responded to faster than if the user has to explicitly check) and social factors (e.g. if I only check my email twice a day).

Where the natural pace of the task does not match the pace of the channel something has to give – this may involve slowing the task, reallocating roles so that less (or more) communication is needed, running several tasks in parallel, and creating communications which foresee likely circumstances. These patterns were predicted largely by a priori argument, but were found to match exactly those found in reported interactions [1,13,18]

2.2 status–event analysis

Status–event analysis is the name for a number of analytic techniques, some formal, some informal, based around the differentiation of phenomena into events – things that happen at a particular moment; and status – things that have some persistent value or state [4,7].

Examples of the former include a mouse click or keystroke, a warning 'beep' from the computer or the arrival of a letter in the post. Examples of status phenomena include the location of the mouse, the current screen contents or the weather.

Although based on a very simple ontological distinction, status–event analysis is powerful because the patterns of interaction between status and event phenomena are found at all levels of a system including human–human, computer–computer and human–computer interactions. Moreover physical phenomena are usually status whereas most computing formalisms focus on event phenomena.

Two particular properties of interest here are the way in which status change gives rise to events (for example, when the temperature of the oven is hot enough for the cake) and the fact that the distinction itself depends on granularity of analysis – and alarm clock ringing is an event in the day as a whole, but a status phenomena when consider the few minutes whilst waking up. These issues of granularity have been a constant problem in the formalisation of temporal phenomena simplistic discretisation has been found to be insufficient [16], but richer treatments, in particular Kutar's work [14], are still partial.

2.3 formal aspects – LADA

In addition, to these more conceptual frameworks I've worked on various more formal treatments including formalisations of status–event analysis [7], the early formal modelling of temporal phenomena in the context of the PIE model [3] and LADA (logic for the analysis of distributed action), which attempts to model actors who are acting independently and then occasionally meeting to take part in joint actions [6].

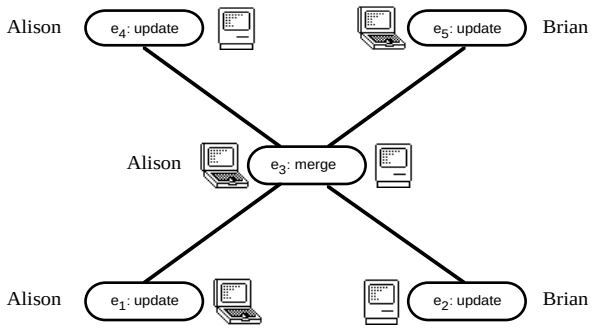

Figure 1. concurrent action in LADA

More recently I have been investigating the way in which different forms of dialogue and task notation map onto simple timelines or traces [9].

3. The task counter

3.1 task sequence

Task and process decomposition notations allow one to discuss the individual sub-tasks or activities within a larger activities and also to discuss the order of those subtasks.

Figure 2 shows a simple process diagram" in the morning I collect post form my pigeon hole, then take it to my desk and finally open (and read) the post. Figure 3 shows an Hierarchical Task Analysis (HTA) for making a cup of tea [19]. here the order of subtasks is more complicated and given by the separate plan. sub-task 2 , "get out cups" comes after boiling the kettle and pouring the tea (task 4) comes after making it (task 3). Task 3 starts only when the kettle boils, not immediately after task 1 or 2. Figure 4 shows a ConcurTaskTree (CTT) diagram for booking a holiday; this uses more formal operators for order derived from LOTOS [15].

Figure 2. simple work process

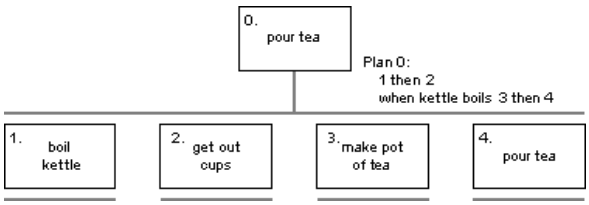

Figure 3. Hierarchical Task Analysis

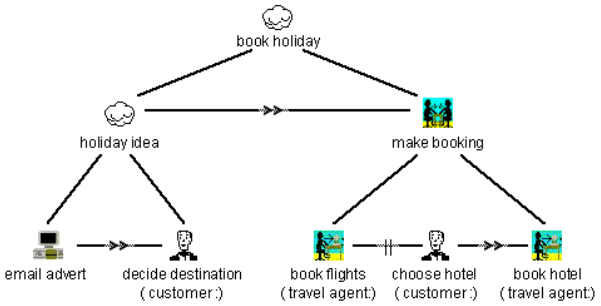

Figure 4. ConcurTaskTree

However, in all these examples, the "followed by" should be read with some care. Although I cannot bring the post to my desk until I have got it from the pigeon hole, there is no guarantee I may not spend time at the pigeon holes chatting to a colleague. Neither the arrow in the process diagram, the "then" in the HTA plan, nor even the ">>" in the CTT say that the next task happens immediately, just that it cannot happen *before* the preceding task and that once the preceding task is complete, the following task will *eventually* occur.

Between the boxes representing the task are gaps. In typography one talks about the "counter", the space between the letters. It is in fact the balance of the counter that gives one the sense of aesthetically well positioned text. Between the boxes representing the gaps are the task counter, the matrix in which the tasks take place. The free running and reliability of tasks rests as much on this counter as it does on the tasks themselves.

3.2 trigger analysis

Whilst the order in the various forms of task analysis say that the next thing will happen, they don't say when it will happen. Trigger analysis is concerned with understanding why tasks happen when they happen. Not just sequence but timing. Actually, this is not even 'just' timing. One might says, for example, that a delivery should be dispatched within 48 hours of an order arriving. However, this timing just expresses the goal of the task – for the task to actually complete within that time, something must happen that drives the pace of the tasks.

Let's look again at the "morning post" task in figure 2. Why do I collect the post from the pigeon hole when I do? Perhaps it is because I always do this first thing in the morning when arriving at work. "Why do I take the post to my desk? Well, having collected it from the pigeon hole it is sitting in my hand, a constant reminder I need to do something with it. Finally why do I open it when I do? Perhaps I do not open it straightaway when I get to my office, but instead put it on my desk to read later, but then mid-morning at coffee time I always read the post.

These things – the routine happenings "every morning first thing"., "during coffee times", and the prompts in the environment "post in my hand" are the triggers that prompt the various tasks to occur. Figure 5 shows the process diagram annotated with these triggers.

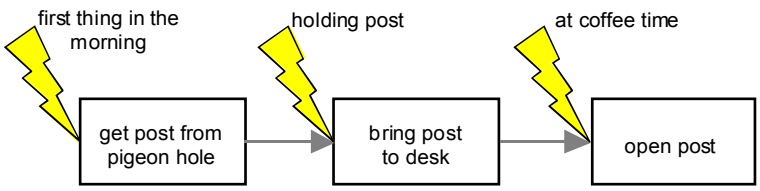

Figure 5. triggers for activities

Trigger analysis both seeks to identify these triggers and also to understand potential failures and other common features of triggers. For example, what happens if there is a fire alarm test one morning disrupting the "at coffee time" trigger ... will the day's post ever get read?

Trigger analysis is discussed in several papers by Devina Ramduny-Ellis, Julie Wilkinson and myself [8,10] and in a chapter in Diaper and Stanton's new Task Analysis collection [12].

3.3 placeholders

As well as knowing *that* something should happen, it is also important to know *what* should happen! This is implicit in a task analysis diagram, but for someone actually doing the task it may be less clear. This is especially a problem if the task is non-routine (e.g. emergency shutdown procedure), drawn out over a long period (e.g. organise a conference), or interspersed with many instances of the same task (e.g. processing an order).

Just as programs require a program counter to say what part of the code to execute, so also humans require what we have called 'placeholders', to record where in a task they are. Sometimes this is purely in the memory of the individuals concerned, sometimes it is explicitly stored in some way, perhaps a checklist or flow chart and sometimes, like triggers, it is stored in the environment in the presence, location and disposition of physical artefacts [10,12].

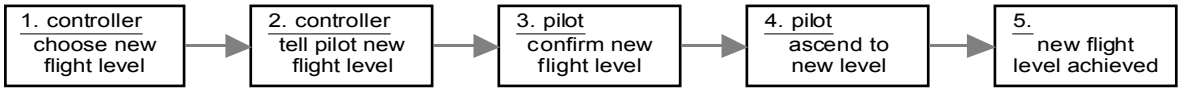

Figure 6. flight level management task

Figure 6 shows a simplified version of the main activities when an air traffic control flight controller wants a pilot to move to a new flight level. When first told about this, I was told that when the controller asks to the pilot to move to a new height he crossed out the old height on the flight strip and wrote in the new one. However, this process is clearly drawn out over several minutes, for example, the pilot may delay acknowledgement if in the middle of a manoeuvre and the plane will have to climb or descend the new height. But there was no obvious placeholder. So I expected that either the controllers had really good memory, or mistakes sometimes happen and the process gets stalled somewhere, or there was some additional cue to act as trigger and placeholder.

When I asked about this I was told that indeed the process is more complicated and that the flight strip is annotated in a more rich manner. Figure 7 shows this process.

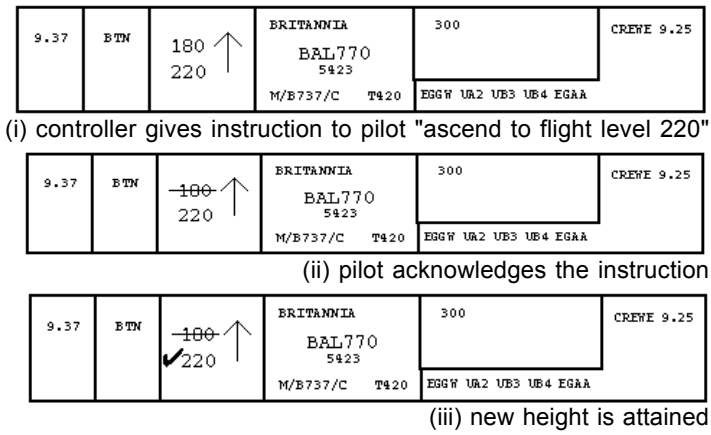

Figure 7. flight strip annotated during task

First of all, when the controller chooses a new level he writes the new level onto the flight strip and an up or down arrow depending on whether it is ascending or descending (i). When the pilot confirms the new level (task 3), the controller crosses out the old level (ii). Finally when the plane has reached the desired height the controller ticks the new level.

The various intermediate states act as cues to the controller both triggering actions and also acting as placeholder. The fact that a new level has been entered, but the old one not crossed out (i) tells the controller that either he has not yet told the pilot or that she has not confirmed the new level. Similarly if the old flight level is crossed out, but the new one not yet ticked it is a placeholder to show that the process is between tasks 3 and 5.

Note that there is no placeholder between tasks 2 and 3 because the recovery for the controller is the same tell or remind the pilot. Also there is none between 4 and 5 because this is the pilot's task not the controllers.

3.4 life history of a task

Recently working with Cristina Chisalita and Gerrit van der Veer [11], we realised that both trigger analysis and groupware task analysis used the ideas of events and triggering of tasks, but in somewhat different ways. In GTA triggering is about the way events create the necessity for a task (e.g. a phone order arriving), whereas in trigger analysis it is the thing (often not directly task related) that makes it happen when it happens.

Looking at these differences realised that the task has three principal 'significant' events. The first is enablement – the thing that makes it that the task should happen at sometime and before which it should not begin. For example, the receipt of an order is the enablement of the order processing task even if the order is put into a tray of pending orders. This is the moment of the triggering event in GTA. The second is initiation – the point at which the task actually starts. For example, the morning coffee break that initiates the opening of post. This is the moment of the trigger analysis trigger event. Finally there is the point at which the task is complete. This is important as it is often the enablement point of the next sub-task.

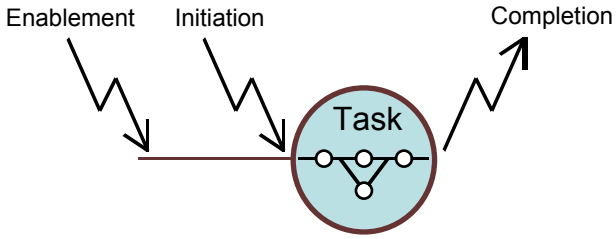

Figure 8. Significant task events

The period between initiation and completion is of course the extent of the task itself. However the gap between enablement and initiation is exactly the task counter.

Having started to look at this counter we begin to see more structure in this gap. The reason that trigger analysis focuses on the initiation point is precisely because there can be delays and interruptions during this gap that can disrupt the task performance. Ethnographic studies and our common experience tells us that such interruptions are the norm not the exception in many circumstances [17]. Also this gap may itself be 'full'. For example, a telephone order may be received, which at one level of analysis is the enablement of the order processing task. However, a closer investigation may show that the person who receives the call writes a note of the order on a post-it note which is later transcribed onto a proper order form that goes into a pending orders tray. That is there a series of activities that comprise the 'gap'.

4. Future work

We are continuing to study the interactions between physical artefacts, events and the detailed timings of tasks. In particular we are aiming to understand the ways in which organisational processes adapt themselves to expected and unexpected interruptions and also the way these interact with meta-events such as the restructuring of organisational roles.

5 Acknowledgements

This work was supported by EPSRC under the DIRC (www.dirc.org.uk) and EQUATOR (www.equator.ac.uk) Interdisciplinary Research Collaborations.

6. References

1. J. Bowers & J. Churcher (1988). Local and global structuring of computer mediated communication: developing linguistic perspectives on CSCW in COSMOS. *CSCW'88 Proc. of the Conf. on Computer-Supported Cooperative Work*, ACM, pp. 125-139.
2. D. Diaper & N. Stanton (Eds.) (2003). *The Handbook of Task Analysis for Human-Computer Interaction*. Lawrence Erlbaum Associates. (in press)
3. A. Dix, The myth of the infinitely fast machine, pages 215-228 in Proceedings of HCI'87: People and Computers III, D. Diaper and R. Winder (eds.), Cambridge University Press, 1987. (Also in A. J. Dix, Formal Methods for Interactive Systems, Academic Press, 1991.)
4. A. Dix (1991). Status and events: static and dynamic properties of interactive systems. Proceedings of the Eurographics Seminar: Formal Methods in Computer Graphics, Ed. D. A. Duce. Marina di Carrara, Italy.
5. A. Dix (1992). Pace and interaction, pages 193-207 in Proceedings of HCI'92: People and Computers VII, A. Monk, D. Diaper and M. Harrison (eds.), Cambridge University Press.
6. A. Dix (1995). LADA - A logic for the analysis of distributed action. Interactive Systems: Design, Specification and Verification,, Ed. F. Paterno. (Proceedings of 1st Eurographics Workshop, Bocca di Magra, Italy, June 1994), Springer-Verlag. pp. 317-332.
7. A. Dix & G. Abowd (1996). Modelling status and event behaviour of interactive systems. *Software Engineering Journal*, 11 (6),. 334-346
8. A. Dix, D. Ramduny and J. Wilkinson (1998). Interaction in the Large. Interacting with *Computers*. **11**(1): 9-32.
9. A. Dix (2002). Towards a Ubiquitous Semantics of Interaction: phenomenology, scenarios and traces. Interactive Systems. Design, Specification, and Verification 9th International Workshop, DSV-IS 2002. P. Forbrig, Q. Limbourg, B. Urban, J. Vanderdonckt (Eds.). Rostock, Germany, June 2002. Springer, LNCS 2545, pp. 238-252
10. A. Dix (2002). Managing the Ecology of Interaction. In *Proceedings of Tamodia 2002 - First Intnl Workshop on Task Models and User Interface Design*, Bucharest, Romania, 18-19 July 2002
11. A. Dix, C. Chisalita and G. van der Veer (2003).Moments of Significance - the meanings of event: enablement, initiation, completion. In Tamodia 2003, part of Universal Access in HCI, Volume 4 of Proceedings of HCI International 2003. C. Stephandis (ed.). Lawrence Erlbaum Associates, 2003. pp. 1519-1523
12. A. Dix, D. Ramduny-Ellis & J. Wilkinson (2003). Trigger Analysis – understanding broken tasks. In Diaper & Stanton (2003).
13. B. Hewitt, N. Gilbert, M. Jirotka & S. Wilbur (1990). *Theories of Multi-Party Interaction*, Social and Computer Sciences Research Group, University of Surrey and Queen Mary and Westfield Colleges, University of London.
14. M. Kutar, C. Britton and C. Nehaniv. Specifiying multiple time granularities in interactive systems. Palanque and Paternó (eds), *DSV-IS 2000 Interactive Systems: Design, Specification and Verification*. LNCS 1946, Springer 2001, pp. 169–190.
15. F. Paternó (1999). *Model-based design and evaluation of interactive applications.* Springer.
16. S. Payne (1993), Understanding Calendar Use, Human-Computer Interaction, 8(2), pp. 83-100.
17. M. Rouncefield, J. Hughes, T. Rodden & S. Viller (1994). Working with "Constant Interruption" CSCW and the Small Office. In *Proceedings of CSCW'94*. Chapel Hill, North Carolina: ACM Press. pp. 275–286.
18. K. Severinson Eklundh (1986). *Dialogue Processes in Computer-Mediated Communication: A Study of Letters in the COM System*, Linkoping Studies in Arts and Science. (excerpt from corpus referenced in Bowers and Churcher 1988)
19. A. Shepherd. *Task analysis as a framework for examining HCI tasks,* in *Perspectives on HCI: Diverse Approaches,* A. Monk and N. Gilbert, Editors. 1995, Academic Press: London. p. 145–174.
20. G. van der Veer & M. van Welie (2000). Task Based GroupWare Design: Putting theory into practice. In *Proceedings of DIS 2000*, New York, United States.
21. M. van Welie (2001). Task-Based User Interface Design. Thesis submitted to the Faculty of Sciences of the Vrije University of Amsterdam