Towards a Ubiquitous Semantics of Interaction: phenomenology, scenarios and traces

Alan Dix

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

Abstract. This paper begins a process of building a semantic framework to link the many diverse interface notations that are used in more formal communities of HCI. The focus in this paper is on scenarios – single traces of user behaviour. These form a point of contact between approaches that embody very different models of interface abstractions or mechanisms. The paper looks first at discrete time models as these are more prevalent and finds that even here there are substantive issues to be addressed, especially concerning the different interpretation of timing that become apparent when you relate behaviour from different models/notations. Ubiquitous interaction, virtual reality and rich media all involve aspects of more continuous interaction and the relevant models are reviewed. Because of their closer match to the real world, they are found to differ less in terms of ontological features of behaviour.

1. Introduction

In the formal HCI literature, notations and modes of description seem to proliferate without limit, both system-oriented dialogue notations and more user-oriented goal and task descriptions. This paper aims to start a process of uncovering common semantic models for user interaction.

There are several reasons for taking on this task:

- practical – so that we can confidently use multiple notations applied to real systems and have a basis for interchange between support tools
- theoretical – so that we can give common semantics to diverse notations, and so understand their overlaps and differences
- philosophical – in grappling with these semantic roots, we begin to have a better grasp of the meaning of interaction

Furthermore, HCI is changing from the 1980's "one man and his machine" to a situation with many people and many devices embedded in a rich environment. Dourish calls this 'embodied interaction' and describes it as "interaction in real time in the real world" [16]. There is comparatively little formal work on these new devices and modes of interaction. Extending existing techniques is going to be a major challenge of the next few years and revisiting our semantic roots, one way to make sense of these multiplying computational phenomena.

So this semantics aims to be ubiquitous in that it is both applicable to:

- notations addressing different aspects of the user interface: task, goal, dialogue behaviour, system state, informal and formal
- types of interaction: GUI, wearable, mobile, ubiquitous, continuous and discrete

In saying "start a process of uncovering common semantic models" this is not because there are no existing semantic frameworks. Indeed, many formal papers have been based on semantic models rather than notations (e.g. the PIE model [11,12], the template model [36], LADA [15], status-event analysis [13,14]). In addition, some notations do have their own semantic models (e.g. trace semantics of process algebras.

However, as a discipline we have few common points to anchor our diverse activities, in contrast to, say, architecture, where the intended physical building links diverse representations (scale models, service schematics, artist's impressions, plans).

This paper proposes that such anchor points are valuable and starts an explicit process of addressing the resulting agenda.

To some extent both the original PIE model and, in a different way, the syndetic modelling framework [6] purport to be universal models of interaction. In this work the aim is to be less normative and more inclusive, setting up a framework for linking multi-paradigmatic analyses rather than proposing a single multi-faceted analysis.

This is potentially a huge task, so this paper does not attempt to address the whole question, but focuses on a phenomenological semantics of scenarios and traces of activity. Observable phenomena are often the easiest point of contact between different descriptions (as in the case of architecture). Furthermore, they are the point of contact between systems and people and so can relate user and system descriptions.

The next section revisits this rationale in greater detail, looking at why diverse notations exist (and should exist), the need for common semantics and the power and complexity of scenarios. Section 3, looks at discrete traces of activity, as is most common in more formal approaches such as STNs, process algebras, grammars and Petri nets. Section 4, considers continuous phenomena, which have more complex temporal behaviour and cannot be characterised as easily in simple state transition schemes. In both cases we will be interested in the way that different levels and different kinds of description may be related to one another, including the relationship between discrete and continuous representations. Finally, we return to the wider agenda and further areas required for a common semantic framework.

2. Rationale

2.1 Bewildering Diversity

There is something about computing that encourages the proliferation of notations and the more formal the area the more different notations we find. This is certainly true in the more formal aspects of user interaction. There are good reasons for this:

- we need to state things precisely, whether we are talking about system states or user tasks, hence notations
- we have different concerns and so want to discuss easily various significant aspects, hence diversity

However, there are downsides:

- a danger of focusing on notations themselves as significant ('I can do this with mine' papers)
- a confusing plethora of incompatible notations understood only by their particular cognoscenti

The latter problem is especially clear in systems specification notations (pick up a previous DSVIS proceedings – do any two papers share a notation?), because they tend to be more detailed and differ in the formal expression of those details. In contrast user-focused notations (e.g. task/goal descriptions) often have 'fuzzy' boxes at the lower levels and allow annotation for complex interactions, making them easier to understand and employ by those outside the notation's cabal.

So, despite the strengths of diversity, there are problems both within the community who accept the importance of formalisation and even more so outside.

2.2 Common Semantics

This paper aims to address these problems in part by starting the process of producing a common semantic framework for user interaction notations.

Note this is not a 'unified notation' or even 'unified method' involving multiple notations, but instead a means of understanding and relating existing and new notations. Neither does this paper contain an extensive review of UI notations, although a common semantics certainly aids the process of comparing and selecting appropriate notations.

Semantic models have proved invaluable in many areas of computing. For example, denotational semantics not only produced a common framework for expressing semantics of programming languages, but lead to a common vocabulary for describing them (binding environment, continuations etc.). Although not usually linked to the theory, we see a similar 'coming together' at a practical level when modules written in different programming languages are compiled into linkable object files – a common semantic form – allowing multi-language programming.

The need for this common semantics is evident in several areas of UI modelling. For example, CTT and ICO have been linked using scenarios [25] which effectively established a *de facto* common semantics between them. Also in model checking, researchers always find themselves manipulating the UI notation to translate it into the notation for the model checker – is this the 'right' manipulation, would a different 'translation' give different results? As interaction becomes more rich and diverse the need for common underpinnings will increase.

Although semantic models are in some ways more abstract than notations, they are often easier to 'ground' in reality and easier to reach common agreement. This is because notations have many concerns over and above 'meaning': tractability, clarity of expression, maintenance etc.

Although we look for a common semantic framework, this does not mean a single unified model. Instead we will find a collection of complementary descriptions that map on to one another, following the pattern of many mathematical formalisms, with several complementary descriptions for 'the same' type of thing. For example, in topology we can start off with open and closed sets or with 'neighbourhoods' of points and from each derive the other. There is even a 'pointless topology' that has neighbourhoods as the primary objects and then 'recovers' points.

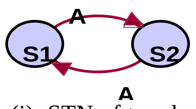
This pattern of multiple related formulations exists within several specific methods. Cognitive complexity theory [20] had a production rule description of goal driven user activity and a system description and verified that the goals can be achieved using the system. This was possible because the lowest levels of goal description involved explicit user actions which could be mapped to system commands. In Abowd's thesis [4] he used both an internal CSP-like process algebra description and also an external 'set of traces' description. These were not duals of one another, but instead specified different aspects of the required behaviour. The overall agent behaviour was constrained to satisfy both requirements. Again the linkage was possible because both descriptions share a common event vocabulary. In syndetic theory different interactors are described in their own 'microtheories' then linked by 'macrotheory' [6].

The linking between different semantic models is complicated by the fact that the concrete semantics of a description in a particular notation is governed not just by the formal semantics (where it exists) of the notation, but also the 'binding' of the constructs in the description to phenomena in the real world.

For example, consider, a two state toggle (fig. 1.i) with two states, S1 and S2, and a single action, A, that moves back and forth between the states. This has a very clear formal semantics (fig. 1.ii shows one behaviour assuming start state is S1).

However, it makes a big difference whether we 'bind' this to the on-off state of a computer (S1 = off, S2=on, A=on/off button). In this case all other activity in the computer is enabled/disabled by this state. In contrast, if could 'bind' this to a bold character format tick box in a word-processor dialogue box (S1=bold-off, S2=bold-on, A=click the tick box). The network is only enabled while the dialogue box is visible and may be operated concurrently with other parts of the interface.

Some of this binding may be specified within the formalism, but some level of binding is always outside the formal system itself.



(i) STN of toggle

S1–A–S2–A –S1–A–S2-A–S1 . . .

(ii) semantic behaviour

Fig. 1. Formal semantics of two state toggle

2.3 Scenarios and Phenomena

Although notations differ markedly in the way in which they describe and structure interaction, they 'come together' at the surface level of user and system actions. This is inevitable as in the end an expert in the notation should be able to look at an actual trace of observable interface behaviour and say ‘yes’ or ‘no’ that is or is not valid with respect to this specification. In addition, internal structural elements, such as states, goals, tasks, internal events, can often be associated with points or portions of the trace –'this happens then'.

This use of scenarios as lingua franca can be seen in various places. In the CTTE tool, rich scenarios are mapped onto ConcurTaskTrees (CTT) [31], which are then used to generate formal traces of actions to validate ICO user interface specifications in PetShop [25]. Model checking is used on several UI notations including LOTOS and CTT [28,30], propositional production systems [2] and State Charts [22]. The advantage of model checkers over theorem proovers is that when they fail they give a counter example, in the form of a trace of actions – a scenario.

Looking further than UI notations and formalisms, scenarios have been used as a common focus for diverse design techniques [8], in the form of stories, incidents or snippets of interaction, they are part of the common language of more sociological descriptions of human behaviour, and several 'soft' descriptions such as patterns or cases involve some sort of scenario. So, focusing on 'what happened' or 'what might happen' also puts us in a better position to relate to this level of description.

Note that the focus on phenomenological semantics is not because it is the only level of valid description, as has become a common philosophical position in many areas of HCI. Instead, it is the focus as a practical point of 'agreement' between diverse descriptions, including those involving motivation, intentions and cognitive process of human agents and the mechanisms and abstractions of computational components ... and even descriptions which deny the validity of either!

3. Discrete Behaviour

This section is about discrete behaviour in the sense that it takes place in discrete time. Whether the value domains for phenomena are discrete or continuous is not really relevant as it makes little difference to our models. In continuous time however, we will see that these differences in value domains are more significant.

At first discrete models seem very simpler. They are certainly far more common. In Palanque and Paternó's collection [26] no paper deals with continuous phenomena and in recent DSVIS only three papers in total. However, in some ways discrete models are more complex. The real world exists in continuous time and so we have a touchstone with which to compare our semantics. With discrete behaviours we have to choose which moments we deem significant and the way in which we relate phenomena, which take place over finite times into momentary measures.

3.1 Trace as Event Stream

As a first model of discrete behaviour we'll consider a simple stream of events:

event 1 – event 2 – event 3 ...

This is precisely the model used for grammars such as BNF [35] or TAG [34] and for process algebras such as LOTOS [27] and CSP [3]. For example, given a process:

$P \rightarrow a\ b\ P \mid a\ c\ P$

Then a potential behaviour is:

a – b – a – b – a – c – a – b – . . .

One way to model this is as a function from some time domain Time to events:

trace: Time $\rightarrow$ EventKind { $\times$ Value }

The optional value is to allow for models where the general type of event (e.g. channel in process algebras) may also allow some form of value passing. However, this is not just a feature of process algebras. If we were analysing a trace of user interface events we might say something like "the user clicked the mouse" – this is a recognisable kind of event which has an associated value "at pixel location 517, 323".

Although this 'works' as a representation of event traces, it puts too much emphasis on the arbitrary time domain. Most commonly Time is an interval of the integers, but this suggests uniformity of the time steps which is rarely the case.

An alternative is to tear apart the phenomena and regard a behaviour as a tuple:

Behaviour = <instances, when,what>

where	instances:	set of EventInstance
	when$_<$:	partial order EventInstance $\leftrightarrow$ EventInstance
	what:	EventInstance $\leftrightarrow$ EventKind { $\times$ Value }

The EventInstance is merely a unique label allowing us to distinguish events that have similar external appearances but occur at different instants – such as the different 'a's in the trace above. The order is the weak 'happens before' relationship. Notice how this allows us to separate out when the event happens – the order of events, from what happens – the event kind and possible value.

Both 'when' and 'what' have the obvious constraints:

dom(when) $\cup$ range(when) $\cup$ dom(what) $\subseteq$ instance

Note the partial order rather than a total order as some notations, such as LADA [15] represent truly concurrent, as opposed to interleaved, events.

Optionally, we can then add actual times to these events

at: EventInstance $\rightarrow$ Time

With the temporal constraint (N.B. second order is 'when' order):

$\forall\ ei_1, ei_2 \in$ instances $\bullet$ $at(ei_1) < at(ei_2) \Rightarrow ei_1 < ei_2$

Notice this is carefully phrased we do not have a double implication. Some notations allow several simultaneous but causally ordered events – so we allow instances to be 'at' the same time, but ordered by 'when'. Note also that we can easily transform a time function trace into this more general behaviour.

Finally, although we shall use this model as a touchstone in the rest of this section it is not intended to be a final point, merely of sufficient richness to discuss various issues and flexibility to allow the necessary extensions to produce a truly ubiquitous semantics.

3.2 Relating Behaviours

We can relate behaviours of this kind to one to another by introducing a partial mapping between event instances. That is, Let X and Y be two behaviours and B_{XY} a binding between them:

B_{XY}: partial mapping X.instances $\leftrightarrow$ Y.instances

A relationship of this kind can arise in two ways:

- a particular trace of activity is captured and the actual simultaneity of particular related events in the two models recorded automatically or annotated by hand
- a binding is defined between models, which then gives rise to an induced binding between behaviours of models

The binding mustn't introduce an inconsistent ordering:

$\forall\ xi_1, xi_2 \in$ X.instances, $yi_1, yi_2 \in$ Y.instances $\bullet$

$B_{XY}(xi_1,yi_1) \wedge B_{XY}(xi_2,yi_2) \Rightarrow \neg(xi_1<xi_2 \wedge yi_2<yi_1) \wedge \neg(xi_2<xi_1 \wedge yi_1<yi_2)$

This weak ordering preservation is to allow things such as one event in one model to correspond to several in another.

However that this can lead to some 'interesting' orderings. For example:

X.instance = {a, b, c}, X.when = {a< b, a < c, b < c}

Y.instance = {p, q}, Y.when = {}, B_{XY} = {(a,p), (b,p), (b,q), (c,q)}

Note that this binding allows us to infer that p and q in some way 'overlap'. This suggests that a full rich semantics of scenarios requires 'events' with duration and that both the 'when' relation and the binding between behaviours needs a set of linear interval relationships such as 'starts before', 'overlaps the beginning' as in [1]. We will not expand on the full semantic model for this here, however because of the separation of the instances, order and values, this extension is not problematic.

3.3 Turn-taking and States

Some models, for example the PIE and related models [12] and action–effect rules [23] have a series of user-action – system-response turns. In the PIE this is modelled as an interpretation function (I) between traces of user commands (from P = seq C) to system effects (E):

I: seq C $\rightarrow$ E

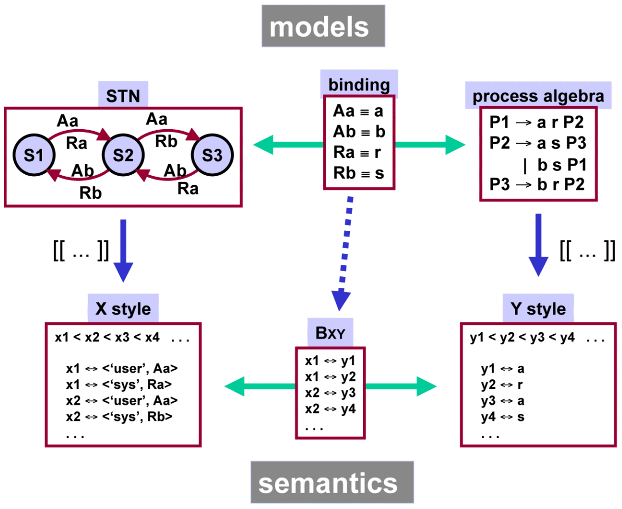


Fig. 2. Behaviour bindings derived from model binding

The unfolding of the system can then be thought of as a turn-taking sequence:

$$e_0 - c_1 - e_1 - c_2 - e_2 - c_3 - e_3 - c_4 - e_4 - ...$$

where: $e_0 = I(<>)$, $e_1 = I(<c_1>)$, $e_2 = I(<c_1,c_2>)$, ...

State-transition diagrams and related formalisms have states with transitions labelled by the action that caused them and any system response. that is the formal model they represent is:

$$\text{trans: (old) State} \times \text{Action} \rightarrow \text{Response} \times \text{(new) State}$$

This gives rise to sequences of the form:

$$s_0 - a_1 - r_1 - s_1 - a_2 - r_2 - s_2 - a_3 - r_3 - s_3 - ...$$

where: s_0 = initial state, $r_i, s_i = \text{trans}(a_i, s_{i-1})$

Note, a new state may be reached 'before' the response, but persists hence the trace order. If we look only at the externally observable phenomena, this then reduces to an action–response trace

$$a_1 - r_1 - a_2 - r_2 - a_3 - r_3 - a_4 - r_4 - \ ...$$

This can be viewed as an event behaviour in two ways.

The first way, which is probably the most natural semantics for the STN is to regard the action and response as being facets of the same event.

$$\text{X.instances} = \{x_1, x_2, x_3, x_4, ...\} \quad \text{X.when} = \{x_1 < x_2 < x_3 < x_4 < ...\}$$

$$\text{X.what} = \{ (x_1,\text{'user'},a_1), (x_1,\text{'sys'},r_1), (x_2,\text{'user'},a_2)\ (x_2,\text{'sys'},r_2), \ ... \}$$

The second is to regard the action and response as separate events, which is what we would get if we 'translated' the STN into an 'equivalent' process algebra specification.

$$\text{Y.instances} = \{y_1, y_2, y_3, y_4, ...\} \quad \text{Y.when} = \{y_1 < y_2 < y_3 < y_4 < ...\}$$

$$\text{Y.what} = \{ (y_1,\text{'user'},a_1), (y_2,\text{'sys'},r_1), (y_3,\text{'user'},a_2), (y_4,\text{'sys'},r_2), \ ... \}$$

Happily these two semantics can be given a binding, so that they can be related to one another in the obvious way:

B_{XY}: $= \{ (x_1, y_1), (x_1, y_2), (x_2, y_3), (x_2, y_4), (x_3, y_5), (x_3, y_6), \ \ldots \ \}$

If the process algebra had been derived from the STN, or if we were attempting to demonstrate equivalence, we would have some sort of binding relating the STN as a model and the process algebra as a model. In such cases the binding between behaviours would be able to be derived from the model binding (fig. 2).

3.4 Interstitial Behaviour

If we choose to retain the states in our semantics of the STN we would get a third semantics obtained from the X semantics by inserting extra instances between the action-response instances:

Z.instances $= \{z_1, z_2, z_3, z_4, \ldots\}$, Z.when $= \{ z_1 < z_2 < z_3 < z_4 < \ldots \}$

Z.what $= \{$ $(z_1,\text{'state'},s_0)$, $(z_2,\text{'user'},a_1)$, $(z_2,\text{'sys'},r_1)$, $(z_3,\text{'state'},s_1)$, $(z_4,\text{'user'},a_2)$, $(z_4,\text{'sys'},r_2)$, $(z_3,\text{'state'},s_2)$, $\ldots$ $\}$

We can relate this semantics to the X semantics:

B_{XZ}: $= \{ (x_1, z_2), (x_2, z_4), (x_3, z_6), (x_4, z_8), \ \ldots \ \}$

Notice that in the relationship between the X and Y semantics we have several instances in the Y behaviour mapping onto a single instance in the X behaviour. In this case, some of the instances in the Z behaviour have no correspondence at all in the X behaviour, they fall in the gaps between events – the interstice (fig. 3).

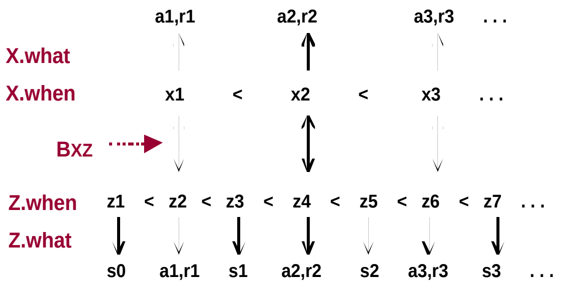


Fig. 3. Binding between X and Z semantics 'leaves out' states

There are two reasons for instances in one behaviour to fall in the interstice of the other. Consider again a process algebra used to specify the interface. Some of the events would correspond to user actions and observable system responses, others would be purely internal. If we wanted to relate the full trace of events with the externally observable one, the hidden internal events would fall in the interstice between the user actions and system responses. These correspond to specific event that happen between the observable events, a form of trace refinement.

In contrast, states are not things that 'happen', but represent something that has a value *throughout* the interval between the events. That is there is a fundamental ontological distinction between the STN states and events. The former are status phenomena – things which have a continuing value over a period of time as opposed to events which occur at an instant. This status–event distinction and the importance of interstitial behaviour was originally proposed in order to deal with continuous behaviour such as mouse movement [13], but as we see, it makes a difference even to the way we regard discrete phenomena.

3.5 Hierarchical and Layered Models

It can be useful to consider some of the internal or structural parts of a model as 'observable' phenomena as we have shown with STN states or hidden events.

Many task or cognitive models are based on hierarchies (e.g. GOMS [7], HTA [37], CTT [29]) and traces of unit actions form a bridge with internal system descriptions. It is also good to represent the higher-level tasks or goals, which require a more flexible model of instances, as discussed in section 3.2, because higher-level tasks 'happen' over a period incorporating several lower-level tasks.

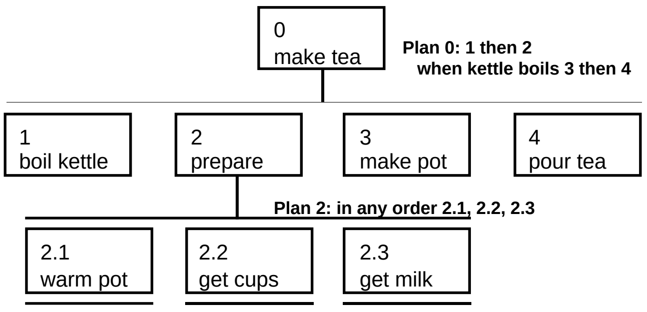


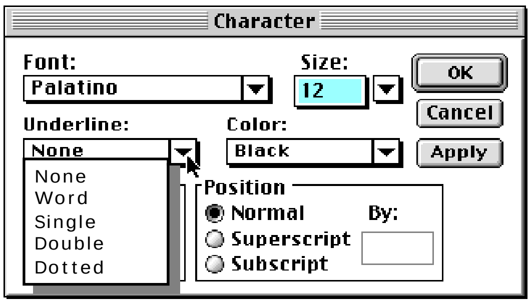
Fig. 4. HTA of tea making

For example, consider the HTA in fig. 4. Consider now a possible behaviour, call it K, for this task in which the user accidentally adds tea to the pot before warming it:

K.instances = { k1, k2, k3, k4, k5, k6, k7, . . . }
K.when = { k1 contains k2, k1 contains k3, k1 contains k4, . . .
k2 before k3, k3 before k4, ... k3 contains k9, k3 contains k10, …
k9 before k10, k10 before k11, . . . }
K.what = { k1 ↔ <'task',0>, k2 ↔ <'task',1>, k3 ↔ <'task',2>, . . .
k9 ↔ <'task',2.2>, k10 ↔ <'task',2.1>, . . . }

Notice how k3 is an instance of the higher-level task "2. prepare", which persists over an interval containing k9, k10 and k11 (the sub-tasks 2.2, 2.1 and 2.3).

Slightly different considerations apply to layered architectural models such as Seeheim [33], Arch-Slinky [39] or PAC hierarchies [9].



Font: Palatino
Size: 12
Underline: None
Color: Black
Position: Normal

Fig. 5. Simple dialogue box

Consider the dialogue box in fig. 5. The user interacts with the pull-down menu in series of lexical-level events, managed by the platform widget toolkit. Only at the end of this series of lexical events is there a dialogue level event 'new underline style is word'. There may then follow further lexical and dialogue events (select position, menu selection of size), before finally the user clicks on 'OK' giving rise to an application-level event and the selection format is changed (fig. 6).

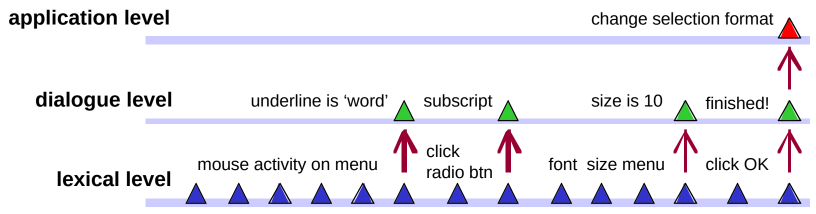


Fig. 6. levels and interaction points

Note the difference. In a task analysis the user is 'selecting the underline mode' <u>throughout</u> the menu interaction, but when considered in architectural terms the dialogue event happens <u>at the end</u> of the menu interaction. Dialogue events can be seen as coinciding or possibly being 'just after' the final lexical level event of the sequence (the mouse release). These points of synchronisation between different architectural levels are called interaction points [17]. We can regard this either as multiple behaviours with bindings between them at the interaction points, or as a single behaviour with event instances at different levels related by 'coincident with' or 'just after' relations. However, this raises another issue.

We have multiple events at an interaction point – e.g. lexical level "release mouse over 'word'" and dialogue level "underline is word". Depending on your viewpoint or concerns these may be validly regarded as coincident or as following closely after one another. The former view would make sense from a user perspective and the latter if we wanted to trace the flow of events through architectural components.

3.6 Temporal Granularity

This problem of 'coincident' vs. 'just after' when considering interaction points is because different models operate at different temporal granularities. Some relations make sense across different granularities – it would be wrong to have event A be before event B in one model and after in another. However, coincidence should always be read as 'at the same time at this model's temporal granularity. This is one of the reasons for the weak link between clock time and order in section 3.1.

The fuzzy nature of human time has been emphasised by Payne in his study of calendar use [32]. In a previous DSVIS paper and her thesis [21], Kutar addressed the complex issue of extending temporal logics to express statements, such as 'on the same day', that have both contextual and fuzzy semantics. This work proposed some solutions but also left many open problems. Even in the simpler world of individual behaviours, problems of temporal granularity are far from trivial, but appear more tractable. Better understanding the meaning of temporal behaviours may give some insights into the more demanding world of temporal specification.

3.7 Real Time

Most models of discrete UI behaviour take place in a task-paced world timed by significant events rather than the ticking of the clock. There are exceptions to this. Temporal extensions to many notations allow one to express the time between events. This can be modelled by the 'at' time discussed in section 3.1, but this forces a global time on everything. For some models it may be more appropriate to use an enrichment of the 'when' relation between events to be able to say things like A happens *sometime* before B and B happens *at least 3 seconds* before C.

The τ-PIE in "The myth of the infinitely fast machine" [10] embodies the external flow of time very directly in the use of 'ticks' to represent moments where there is no user activity. This can be given a semantics in terms of the behaviours in two ways. One way is to use the same 'trick' of instances where the 'what' is either empty or a special 'nothing happens' kind. Alternatively, we can only have instances for actual user actions or system effects and use the 'at' to give real times to events. These two semantics can be mapped to one another, so the choice is a matter of convenience.

4. Continuous Behaviour

We now move on to continuous behaviour. This discussion will be more limited for several reasons: partly because there are fewer models of continuous interaction and partly because the world really is continuous so there are less 'choices' about models. We will also find that many of the issues of continuous interaction have been prefigured by discrete issues, so can be dealt with quite succinctly.

4.1 Models of Continuous Interaction

Although many of the systems and interfaces studied in rich-media and in novel interfaces embody continuous real-time interaction, there are few models of this in the HCI literature. Possibly this is because of the conceptual dominance of discrete models. We start or analysis of continuous interaction by reviewing these models.

Probably the earliest continuous time models in the formal UI literature are the variants of status–event (S–E) analysis [13,14], and we have already seen a hint of this in discrete systems. This distinguishes events, which occur at specific moments of time, from status phenomena, which have (typically changing) values over a period of time. Examples of events include keystrokes, beeps, and the stroke of midnight in Cinderella. Examples of status phenomena include the current display, the location of the mouse pointer, the internal state of the computer and the weather.

Perhaps one of the most significant features of S-E is its treatment of interstitial behaviour. Whereas discrete models focus on the moments when events occur, S-E puts equal emphasis on the more fluid interaction between events. In many GUI systems this is what gives the 'feel' of interaction: dragging, scrolling etc., and in rich media this is likely to be the main purpose of interaction!

Status–event analysis is really a conceptual framework for viewing interaction, but does have several concrete models, both descriptive (variants of the PIE) and specification notations. These all have the general form of a state-transition style description of events and more continuous description of interstitial behaviour:

action: user-event $\times$ (current/history of) input-status $\times$ state
$\rightarrow$ response-event $\times$ (new) state

interstitial behaviour:
(current/history of) input-status $\times$ state $\rightarrow$ output-status

The treatment of the input status at events and during interstitial behaviour distinguishes interactions of markedly different kinds: trajectory independent interactions that only depend on the current state (such as where the mouse button is when it is clicked) vs. trajectory dependent interactions where the complete status history matters (such as freehand drawing).

Another crucial aspect of S–E, not apparent in discrete system, are status-change events. These occur when a status phenomena crosses some trigger threshold, for example, a target temperature or particular time (on the clock). The nature of threshold is application dependent and may be dynamic. Furthermore there issues concerning how status–change events become system-level events: polling, active sensors, etc. At an abstract level of specification one would just say "when this happens ...", but as this becomes operationalised issues of mechanism surface.

In modelling virtual reality systems, Wüthrich [41] made use of cybernetic systems theory using formulae of the form:

$$\text{state}_t \;=\; \phi\,(\,t,\, t_0,\;\; \text{state}_{t0},\;\; \text{inputs during } [t_0,t)\,)$$

$$\text{output}_t \;=\; \eta\,(\,\text{state}_t\,)$$

together with appropriate consistency conditions.

The difference between this and the S–E description is largely representational, although the ‘state’ is different. In the S-E description the 'state' ignores ephemeral changes during the interstices, whereas the systems theory model incorporates this (for example, that the mouse is at a particular position *now*). Both forms of state can be useful and can be incorporated together as alternative views or levels of description.

As a personal preference I regard the systems theory form as a potential replacement for the interstitial behaviour, but would retain the distinct action transition as this accords more closely with the way we tend to informally describe systems. However, both forms of expression should be accommodated within a ubiquitous theory.

Notice that the systems theory definition treats events on a par with status phenomena, both are merely part of an 'inputs at time t'. However, when viewed as functions of time, event phenomena have values only at a few points of time. There is a history of dealing with event phenomena in this way in applied mathematics where 'impulse' events, such as a voltage spike or collision, are modelled using Dirac delta functions (infinitely large for infinitesimally short time). In fact, when applying systems theory to a real example Wüthrich finds it more convenient to describe event phenomena as time-stamped values.

The TACIT project [38] studied a variety of continuous time interactive systems and also considered modelling using hybrid high-level Petri-Nets [24]. In common with many models from the hybrid systems community [19], this takes a largely dualist model of the world with discrete computer systems interfacing with a continuous environment defined by differential equations. The formal expression of this involves 'continuous transitions' – a form of interstitial behaviour.

A component/interactor model based on a similar variant of hybrid Petri-nets is proposed in [40] to deal with objects in virtual environments. Its 'world object state' has both discrete and continuous output and input (fig. 7) and a threshold on continuous input, producing status-change events. Less clear from the diagram above, but evident in the Petri-net specifications, is the direct connection between continuous input and output (status–status relations) controlled by the enable/disable switch.

Note that it is quite easy to describe these models in similar terms because they sit closer to the real world than in the discrete case. For behaviour modelling we need only extend the EventInstances to include values that are functions of time (no longer 'EventInstances', just Instances). This can then represent periods where the mouse position moved over a trajectory, or the window tracked the mouse position.

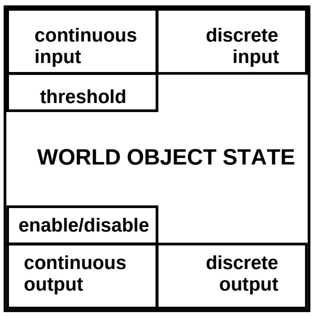


Fig. 7. Continuous interactor from [40]

4.2 Granularity and Two Timing

Granularity effects resurface in continuous time and are perhaps more perplexing. We still want to be able to talk about events: the button was clicked, the computer beeps, I had a meeting. It is right to represent these as events, but clearly when looked at closely they take some duration. We can allow many-to-one bindings just as we do in the discrete case. However, this appears a little odd where 'many' is a period with continuous motion (for example, if select menu item at one level became the actual mouse movements at another). In fact, this is not uncommon in applied mathematics where solutions of equations with slow time-varying parameters or small non-linearities are managed by imagining two timescales, one infinitely fast with respect to the other. A technique called two-timing.

A less drastic approach, which captures the finite difference in granularity, is to have bindings which both preserve order and also have bounded 'jitter'. That is, there is a maximum 'jitter' time, δ, such that if event A and B are separated by time t in one behaviour their separation in the other behaviour is in the range $[t-\delta, t+\delta]$.

4.3 Temporal Gestalt Phenomena

The issues of hierarchy and architectural levels are similar in the continuous case to the discrete one. However, continuous media exhibit a special case of the lexical–dialogue interaction points – temporal gestalt phenomena. Consider gesture recognition – the movement only has meaning once completed, music – the melody is only a melody whilst being played, and even graphics get meaning through movement [5]. This is rather like the fact the typed characters "d", "i", "r", return only mean "list directory" when considered as a whole, but the boundaries are clearer in the case of discrete phenomena – the gesture has no definitive start or end although an endpoint would be operationalised by actual recognition software.

5. Summary

This paper has considered the semantics of traces of user-system activities in both discrete and continuous interaction. One might have thought that this would have been a trivial task as all models in some sense are describing 'the same thing', but as we have seen the multiplicity of viewpoints and levels of analysis mean that even establishing commonality at the level of sequences of actions is surprisingly complex. The paper does not present a final or complete semantics, but does demonstrate how common behavioural semantics may be possible for a range of common notations.

We have also seen a number of problematic issues for formal treatment of scenarios, especially when we try to align scenarios corresponding to different complementary models. This is because notations differ in which events and actions they regard as atomic, or instantaneous and much of our discussion has been about how to establish bindings between these different accounts.

Another issue that has arisen in both discrete and continuous models is the ontological distinction between status and event phenomena and the important role of interstitial behaviour.

The word 'theory' comes from two Greek words *thea*, "outward appearance", and *horao*, "to look closely" [18]. This paper has been examining the surface of interaction and it is hoped that by following this through in detail we can work towards a theory of multi-paradigm interaction modelling.

Acknowledgements

This work was supported by the EPSRC funded project DIRC.

References

1. G. Abowd. *Formal aspects of human computer interaction*, PhD Thesis, University of Oxford, 1991.
2. G. Abowd, H. Wang, and A. Monk. *A formal technique for automated dialogue development*. in Proc. of DIS'95. ACM Press, (1995). p. 219–226.
3. H. Alexander, *Formally-based tools and techniques for human-computer dialogues*. Ellis Horwood, (1987).
4. J. Allen. *Planning as Temporal Reasoning*. In Proc., 2nd Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, (1991).
5. M. Bacigalupi. Designing movement in interactive multimedia: making it meaningful. *Interfaces* **44**, Autumn (2000), pp. 12-15.
6. P. Barnard, J. May, D. Duke and D. Duce. *Systems, Interactions, and Macrotheory*. ACM Transactions on Computer-Human Interaction, (2000). **7**(2):222–262.
7. S. Card, T. Moran, and A. Newell. *The Psychology of Human Computer Interaction*. 1983, Lawrence Erlbaum. (1993).
8. J. Carroll (ed). *Scenario-Based Design: envisioning work and technology in system development*. Wiley, (1995).
9. J. Coutaz. *PAC, an object oriented model for dialogue design*. In Proc. INTERACT'87. H-J Bullinger B. Shackel (eds). Elsevier (North- Holland), (1987) pp 431-436.
10. A. Dix. *The myth of the infinitely fast machine*. People and Computers III - Proceedings of HCI'87, D. Diaper and R. Winder (eds.). Cambridge Univ. Press, (1987) pp. 215-228.
11. A. Dix and C. Runciman. *Abstract models of interactive systems*. in People and Computers: Designing the Interface. Cambridge University Press, (1985) pp. 13-22.
12. A. Dix. *Formal Methods for Interactive Systems*. Academic Press, (1991).
13. A. Dix. *Status and events: static and dynamic properties of interactive systems*. in Proceedings of the Eurographics Seminar: Formal Methods in Computer Graphics. Marina di Carrara, Italy. (1991).
14. A. Dix and G. Abowd, *Modelling status and event behaviour of interactive systems*. Software Engineering Journal, (1996) **11**(6):334–346.
15. A. Dix. *LADA — A logic for the analysis of distributed action*, in Interactive Systems: Design, Specification and Verification (1st Eurographics Workshop, Bocca di Magra, Italy, June 1994), F. Paternó (ed). Springer Verlag, (1995) pp. 317-332.
16. P. Dourish. *Embodied Interaction*. ACM Press, (2001).
17. C. Gram and G. Cockton (eds.). *Design Principles for Interactive Software*. Chapman and Hall, 1996.
18. D. Gregory. *Geographical Imaginations*. Blackwell, (1994).
19. R. Grossman *et al.* (eds). *Hybrid Systems*. LNCS 736, Springer Verlag, (1993).
20. D. Kieras and P. Polson, *An approach to the formal analysis of user complexity*. International Journal of Man-Machine Studies, (1985). **22**:365-394.
21. M. Kutar, C. Britton and C. Nehaniv. *Specifiying multiple time granularities in interactive systems*. In DSV-IS 2000 Interactive Systems: Design, Specification and Verification. P. Palanque and F. Paternó (eds). LNCS 1946, Springer, (2001) pp. 169–190.

22. K. Loer and M. Harrison. *Formal interactive systems analysis and usability inspection methods: two incompatible worlds?* In DSV-IS 2000 Interactive Systems: Design, Specification and Verification. P. Palanque and F. Paternó (eds). LNCS 1946, Springer, (2001) pp. 169–190.
23. A. Monk. *Action-effect rules: a technique for evaluating an informal specification against principles.* Behaviour & Information Technology, (1990) **9**(2):147-155.
24. M. Massink, D. Duke and S. Smith. *Towards hybrid interface specification for virtual environments*. In DSV-IS 1999 – Design, Specification and Verification of Interactive Systems. D. Duke and A. Puerta (eds). Springer, (1999) pp. 30–51.
25. D. Navarre. P. Palanque, F. Paternó, C. Santoro and R. Bastide. *A tool suite for integrating task and system models through scenarios*. In DSV-IS 2001 Interactive Systems: Design, Specification and Verification. C. Johnson (ed). LNCS 2220, Springer, (2001) pp. 88–113.
26. P. Palanque and F. Paternó (eds). *Formal Methods in Human Computer Interaction*. Springer-Verlag, (1997).
27. F. Paternó and G. Faconti. *On the use of LOTOS to describe graphical interaction*. in Proceedings of HCI'92: People and Computers VII. Cambridge University Press, 1992. p. 155–173.
28. F. Paternó. *Formal reasoning about dialogue properties with automatic support*. Interacting with Computers. August (1997) pp. 173–196.
29. F. Paternó. *Model-based design and evaluation of interactive applications.* Springer, 1999.
30. F. Paternó and C. Santoro. *Integrating model checking and HCI tools to help designers verify user interface properties*. In DSV-IS 2000 Interactive Systems: Design, Specification and Verification. P. Palanque and F. Paternó (eds). LNCS 1946, Springer, (2001) pp. 135–150.
31. F. Paternó, G. Mori and R. Galimberti. *CTTE: an environment for analysis and development of task models of cooperative applications.* In Proceedings of CHI'01, Vol 2, ACM Press, (2001).
32. S. Payne. *Understanding Calendar Use.* Human-Computer Int., (1993) **8**(2):83-100.
33. G. Pfaff, and P. Hagen (eds). *Seeheim Workshop on User Interface Management Systems*. Springer-Verlag, (1985).
34. S. Payne and T. Green, *Task action grammars: a model of mental representation of task language.* Human–Computer Interaction, (1986), **2**(2):93–133.
35. P. Reisner. *Formal grammar and human factors design of an interactive graphics system*. IEE Transactions on Software Engineering, 1981. **7**(2):229–240.
36. C. Roast and J. Siddiqi. *Using the template model to analyse directory visualisation*. Interacting with Computers. 1997, **9**(2):155-172.
37. A. Shepherd. *Task analysis as a framework for examining HCI tasks,* in *Perspectives on HCI: Diverse Approaches,* A. Monk and N. Gilbert (eds). Academic Press, (1995) pp. 145–174.
38. TACIT: *Theory and Applications of Continuous Interaction Techniques*, EU TMR Network ERB FMRX CT97 0133. http://kazan.cnuce.cnr.it/TACIT/TACIThome.html
39. *The UIMS tool developers workshop: A metamodel for the runtime architecture of an interactive system*. In SIGCHI Bulletin, (1992) **24**(1):32-37.
40. J. Willans and M. Harrison. *Verifying the behaviour of virtual world objects*. In DSV-IS 2000 Interactive Systems: Design, Specification and Verification. P. Palanque and F. Paternó (eds). LNCS 1946, Springer, (2001) pp. 65–77.
41. C. Wûthrich. *An analysis and model of 3D interaction methods and devices for virtual reality.* In DSV-IS 1999 Design, Specification and Verification of Interactive Systems. D. Duke and A. Puerta (eds). Springer, (1999) pp. 18–29.