



HAL
open science

Physigrams: modelling devices for natural interaction

Alan Dix, Masitah Ghazali, Steve Gill, Joanna Hare, Devina Ramduny-Ellis

► **To cite this version:**

Alan Dix, Masitah Ghazali, Steve Gill, Joanna Hare, Devina Ramduny-Ellis. Physigrams: modelling devices for natural interaction. *Formal Aspects of Computing*, 2008, 21 (6), pp.613-641. 10.1007/s00165-008-0099-y . hal-00534919

HAL Id: hal-00534919

<https://hal.science/hal-00534919v1>

Submitted on 11 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Physigrams: modelling devices for natural interaction

Alan Dix¹, Masitah Ghazali², Steve Gill³, Joanna Hare³ and Devina Ramduny-Ellis¹

¹ Computing Department, InfoLab21, Lancaster University, Lancaster LA9 5EZ, UK.

E-mail: alan@hcibook.com

² Department of Software Engineering, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia

³ Cardiff School of Art and Design, UWIC, Cardiff, Wales, UK

Abstract. This paper explores the formal specification of the physical behaviour of devices ‘unplugged’ from their digital effects. By doing this we seek to better understand the nature of physical interaction and the way this can be exploited to improve the design of hybrid devices with both physical and digital features. We use modified state transition networks of the physical behaviour, which we call physiograms, and link these to parallel diagrams of the digital state. These are used to describe a number of features of physical interaction exposed by previous work and relevant properties expressed using a formal semantics of the diagrams. As well as being an analytic tool, the physiograms have been used in a case study where product designers used and adapted them as part of the design process.

Keywords: Physicality; Interaction modelling; Affordance; Natural interaction; Physical devices; Product design; Physiograms

1. Introduction

1.1. Motivation: understanding physical devices

For some years the authors have been interested in understanding what makes some interactions with physical devices seem ‘natural’ whilst others need to be carefully learnt. Part of this lies in the fact that interacting with ordinary objects in the physical world is natural to us; even as babies we reach out, explore our own hands, touch our mothers’ faces, then play with balls and toys. Many aspects of physical interaction are informed by culture and things we have learnt about the technological world, but much is still common and would be equally natural if a person from two hundred or even two thousand years ago were suddenly transported to the present.

One aspect of our studies has been to try and unpack which properties of physical interaction are essential to make them comprehensible, and which can be relaxed [Dix03, GhD03, GhD05]. We can then use these properties to understand how to make digital interactions more natural, and thus inform several forms of design:

- pure digital interaction on a computer screen (although even this requires physical devices, such as the mouse)
- novel tangible interactions where physical objects are imbued with digital properties (as found in tangible interaction, and ubiquitous computing)
- more mundane devices such as mobile phones or even electric kettles.

A critical method for us has been to analyse in detail commonly used artefacts such as an electric kettle or minidisk controller (already looking very dated!). We assume that when such devices appear easy to learn and use the designers have often embodied, either explicitly or implicitly, their own understanding of what makes interaction natural. So by detailed analysis of these everyday artefacts, we have gradually mined the experience, often tacit, of successful designers and documented this in terms of properties and types of device interaction, some of which we shall use later in this paper (e.g. *exposed state*, *bounce-back*, *compliant interaction*). This analysis and the early stages in developing a diagrammatic notation has been reported previously [GhD03, GhD05] and a later paper introduced a more explicitly formal specification to give a level of semantics to the diagrams and also allow formal statements of some properties [DGR07]. The current paper adds to this picture refining some of the formal expression, engaging in more detailed discussion of its implications, extending the work through a practical case study with product designers, and reflecting on use in novel device design.

1.2. This paper: focus and goals

In this paper we investigate the formal representation of the *interactive behaviour* of physical devices. The core idea is to consider devices ‘unplugged’; that is entirely separate from any digital or other external functionality. When a mobile phone battery has run down, or a light switch is unscrewed from the wall; still they both have interactive physical behaviours: the phone buttons can be pressed, the light switch can be flipped with a finger, even though there is no resulting effect on the phone or light. Note the interaction here is directly with the device not with any digital or electronic behaviour. To represent these ‘unplugged’ devices, we will incrementally augment basic state transition networks (STNs) to enable them to represent increasingly complex physical properties identified in our previous work. We call the resulting diagrams *physigrams*.

Of course, this is not to say the digital effects are not important, indeed the advantage of specifying the physical behaviour is that we can see the extent to which it is consonant with the digital behaviour. We use separate but connected models of physical devices (the physigrams) and of their digital effects, and so are able to analyse the physical aspects of a device and also the extent to which this relates sensibly to the digital aspects.

In some ways this is similar to architectural models, from Seeheim onwards [Pfh85], that separate presentation from functionality, but in these models the different levels are all principally digital, with only Arch/Slinky making an explicit attempt to discuss the physical level of interaction [UIM92]. However, even Arch/Slinky puts physical interaction as an additional (lowest) layer whilst we shall see that physical devices embody aspects of at least dialogue-level interaction. This is also similar to work on linking models of interface and functionality (often inspired by Seeheim), for example Moher et al’s Petri Net-based ‘bridging framework’ for linking models of devices, users, and interfaces [MDB96]. However, these, to our knowledge, all stop short of a physical-level behaviour description.

Producing this formal framework is valuable in drawing out insights about the nature of physical interaction. However, we also believe this diagrammatic representation can be useful during the design process. While formal notations are often described as being for ‘communication’ or ‘understanding’, their value in this respect is rarely subject to empirical studies (with exceptions [Joh96]). Certainly the temptation for a formalist is to add more and more features to their notation in order to make it representationally complete, but often at the expense of clarity. What might have started as a simple notation, often becomes obtuse to all but the notation’s developers. Therefore, in order to understand the potential practical benefit of the approach, this paper also looks at how real designers are able to understand and produce the diagrammatic notation. This is not a rigorous evaluation of the notation in practice; however, as a case study of use, it yields interesting insights into both potential features that may be needed and aspects of the notation that should not change.

This paper focuses on the physical behaviour of devices. However, there are other important aspects of the physical nature of a device including the detailed physical form and layout of the device, its ergonomic aspects, its aesthetics, and its disposition in the environment. However, we have found sufficient insight from initially restricting our scope to device behaviour, especially as this is also the level of representation that is most similar to existing user interface specification. We will revisit this issue and look forward to more comprehensive modelling at the end of this paper.

In summary our *focus* in this paper is the interactive behaviour of physical devices ‘unplugged’. Our *scope* includes models both of this physical behaviour and of the digital behaviour controlled by the device, and also the link between the two. The *goals* of the work are:

understanding: to gain generic insights into the nature of physical interaction. Following the pattern common in formal specification of user interfaces, these insights will often have informal expression once they are identified.

design: to allow designers to specify the physical aspects of novel devices and so help them design more effective interaction.

In the remainder of this section we unpack the way in which physical devices relate to their digital effects on a system’s logical state and the kinds of feedback that occur. Section 2 then reviews some critical related work. Section 3 introduces the modelling approach that is then used in Sects. 4–7, which work step-by-step through a number of example devices and systems of increasing complexity. Through these examples, the paper builds up ways of describing the devices in terms of state diagrams of both the underlying logical functions and also the physical aspects of the device. The state diagrams of the physical aspects (the device ‘unplugged’) are augmented to capture some of the interesting aspects of physical interaction (the *physigrams*). A formal model is incrementally developed that gives more precise semantics to both kinds of diagram and the relationship between them. Having developed the diagrams and formalism to deal with different kinds of existing consumer devices, Sect. 8 describes how two product designers used and adapted the physigrams as part of their exploration of alternative designs for a novel device. Finally, we reflect on the lessons learnt and further work required to obtain a complete model of physical interaction with digital devices that is both formally sound and practically useful.

1.3. Physical devices and feedback

When we use the term physical device in this paper, we are using the word slightly differently than is common. We use it to mean the actual physical button, knob or other controls on their own. For example, a light switch has properties when torn from the wall and unwired—that is the physical device is a switch whether or not it is connected to a light. In the case of a mobile phone think of the phone with its innards removed—you can hold it, press its buttons, etc., whether or not you get any digital feedback. Sometimes the term ‘physical device’ would be used for the phone together with its digital functionality, but by separating the two we aim to understand better the relationship between them.

Feedback is a critical aspect of interaction, both with digital entities and with the physical world, and plays a major role in the theory and practice of usability: effective feedback was one of Shneiderman’s principles of direct manipulation [Shn83] and one of Nielsen’s heuristics [NiM94]; also a substantial issue in the early formal modelling of interactive systems was the specification of various forms of observability [Dix91].

Once we think of the physical device and the digital effects separately, we can look at different ways in which users get feedback from their actions. Consider a mouse button: you feel the button go down, but also see an icon highlight on screen.

Figure 1 shows some of these feedback loops. Unless the user is implanted with a brain-reading device, all interactions with the machine start with some physical action (a). This could include making sounds, but here we will focus on bodily actions such as turning a knob, pressing a button, dragging a mouse. In many cases this physical action will have an effect on the device: the mouse button goes down, or the knob rotates and this gives rise to the most direct physical feedback loop (A) where you feel the movement (c) or see the effect on the physical device (b).

In order for there to be any digital effect on the underlying logical system the changes effected on the device through the user’s physical actions must be sensed (i). For example, a key press causes an electrical connection detected by the keyboard controller. This may give rise to a very immediate feedback associated with the device; for example, a simulated key click or an indicator light on an on/off switch (ii). In some cases this immediate loop (B) may be indistinguishable from actual physical feedback from the device (e.g. force feedback as in the BMW iDrive); in other cases, such as the on/off indicator light, it is clearly not a physical effect, but still proximity in space and immediacy of effect may make it feel like part of the device.

Where the user is not aware of the difference between the feedback intrinsic to the physical device and simulated feedback, we may regard this aspect of loop (B) as part of ‘the device’ and indistinguishable from (A). However, one has to be careful that this really is both instantaneous and reliable. For example, one of the authors often mistypes on his multi-tap mobile phone hitting four instead of three taps for letters such as ‘c’ or ‘i’. After some experimentation it became obvious this was because there was a short delay (a fraction of a second) between pressing a key and the simulated keyclick. The delayed aural feedback was clearly more salient than the

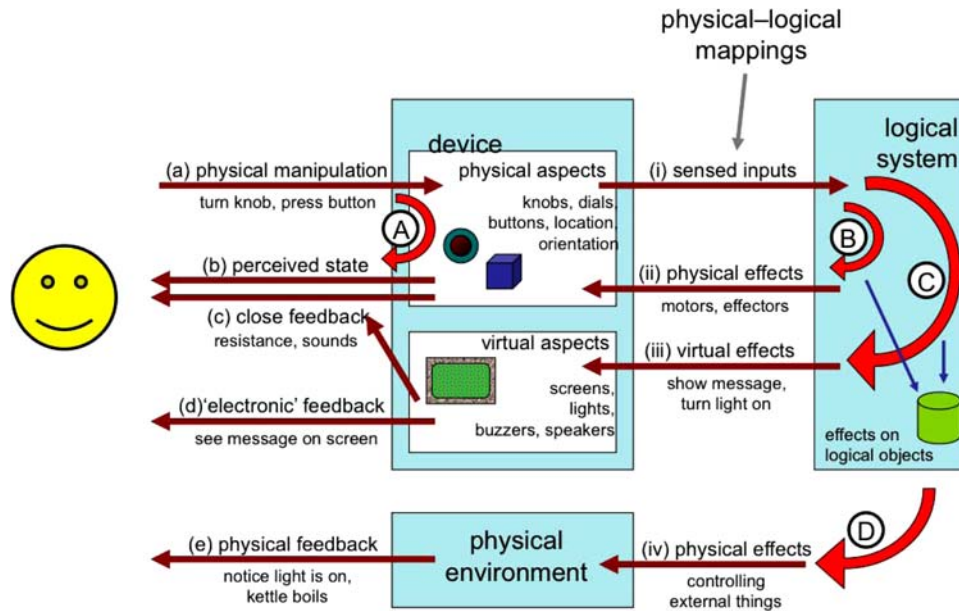


Fig. 1. Multiple feedback loops

felt physical feedback and so interfered with the typing; effectively counting clicks rather than presses. Switching the phone to silent significantly reduced typing errors!

The sensed input (i) will also cause internal effects on the logical system, changing internal state of logical objects; for a GUI interface this may be changed text, for an MP3 player a new track or increased volume. This change to the logical state then often causes a virtual effect (iii) on a visual or audible display; for example an LCD showing the track number (iii). When the user perceives these changes (d) we get a semantic feedback loop (C). In direct manipulation systems the aim is to make this loop so rapid that it feels just like a physical action on the virtual objects.

Finally, some systems affect the physical environment in more radical ways than changing screen content. For example, a washing machine starts to fill with water, or a light goes on. In addition there may be unintended physical feedback, for example, a disk starting up. These physical effects (iv) may then be perceived by the user (e) giving additional semantic feedback and so setting up a fourth feedback loop (D).

For the purposes of this paper we will not care much whether the final semantic effect and feedback is virtual (loop (C)) or physical (loop (D)), nor whether it is deliberate or accidental, as it is the physical device (that is loops (A) and if indistinguishable (B)) in which we are most interested.

2. Related work

2.1. Theories and frameworks for physical interaction

The most obvious connection to this work is Gibson's concept of affordances [Gib86]. For a simple physical object, such as a cup, there is no separate logical state and simple affordances are about the physical manipulations that are possible ((a) in Fig. 1) and the level to which these are understood by the user: Norman's 'real' and perceived affordances [Nor99]. For a more complex, mediated interface the effect on the logical state becomes critical: the speaker dial affords turning but at another level affords changing the volume. Hartson [Har03] introduces a rich vocabulary of different kinds of affordances to deal with some of these mediated interactions. In Sect. 4.2, we will look in detail at Gavers notion of sequential affordance [Gav91].

The 'Sensible, Sensable and Desirable' (SSD) framework [BSK03] deals with this relationship between the physical device and logical state. It considers three aspects of the relationship: sensible—the aspects of the physical device can be sensed or monitored by the system; sensible—the actions that the user might reasonably do to the device; and desirable—the attributes and functionality of the logical system that the user might need to

control. This is used to explore the design space and mismatches between the sensible, sensible and desirable may suggest options for re-design. In Fig. 1, the sensible aspects correspond to (i), whilst the sensible ones refer to possible actions ('real' affordances) of the device (a) that the user might reasonably perform. The desirable part of the framework refers to the internal possibilities of the logical state. Note that what is sensible to do with a device depends partly on perceived affordances and partly on the user's mental model of the device and its associated logical state.

The concept of fluidity, introduced in Dix et al. [DFA04] and expanded in our work leading to this paper, is focused on the way in which this mapping is naturally related to the physical properties of the device. Whereas the SSD framework is primarily concerned with what is *possible* to achieve, fluidity is focused on what is *natural* to achieve. This naturalness involves both cognitive aspects and also more motor-level responses. In this paper we do not explicitly consider the mental models formed by users, but this is implicit in some of the discussion of mappings. However, ongoing empirical work by the authors is aimed at teasing apart cognitive and bodily responses in physical interaction.

The work that is closest in concept to our own is Interaction Frogger [WDO04]. This discusses three kinds of feedback: inherent feedback, augmented feedback and functional feedback, which correspond almost exactly to the loops (A), (B) and (C) respectively. Physical feedback in the environment (loop (D)) is not explicitly described in their work, but would presumably fall under functional feedback. As well as feedback, the Frogger work looks at *feedforward* against each loop, where feedforward is, rather like Norman's perceived affordance, about the different ways a device/system can expose its action potential. Critically too, this work, like our own, is interested in what makes interaction natural and brings out particular qualities that impact this: time, location, direction, dynamics, modality and expression.

2.2. Device abstraction

The central focus of this paper is the detailed modelling of the physical behaviour of interaction devices themselves, with a secondary focus on the way this then links to digital functionality. To some extent this runs completely counter to the long-standing paradigm of device abstraction within user interface specification and construction.

The roots of device abstraction go back a long way, certainly to early graphics standards such as GKS and PHIGS, and were instrumental in allowing the development of applications independent of particular vendors and physical devices, indeed it is hard to envisage the modern GUI without the key abstraction of text input vs. pointer. This basic separation is evident today in the APIs for most user interface toolkits, for example, Java AWT has 'mouse' events even though the actual device is often a trackpad or stylus.

Card et al.'s Design Space for Input Devices [CMR90, CMR91] was influential in developing these notions beyond simple 2D pointers. Their analysis is particularly relevant to our work as they used as running example a radio with knobs and dials—defining these and general input devices in terms of primitive dimensions, many of which have become common language: relative/absolute, position/force, linear/rotary. Their work is also interesting in that, on the one hand, it abstracted devices into classes, but, on the other, it took into account that setting a value through rotating a dial (rotary) is different from moving a slider (linear)—that is, at least certain aspects of the physical nature of the device are important.

Device abstraction has clearly been vital to the ongoing development of interface software and hardware allowing software to be written once irrespective of the intended device and allowing new hardware, such as the trackpad, to be deployed without needing to modify or port existing applications. This is problematic in multi-modal interfaces, where the difference, for example, between speech or keyboard entry of text is intrinsic to the domain. But here too, there have been long standing efforts to add layers of abstraction, for example, in the PAC-AMODEUS architecture [NC095].

While clearly valuable, the drive to device abstraction does elide differences that may be important, for example, the fact that a mouse button mounted on the top of a mouse may be difficult to hold down when lifting the mouse during long drag actions, whereas the equivalent interaction would be fine using a velocity joystick or inward facing mouse buttons (now rare!).

Recognition of the importance of these device differences can also be tracked to early days of HCI as a counter trend to device abstraction. It has long been recognised that devices that are functionally similar but physically different also differ in performance typically measured using Fitts' Law constants (e.g. reviews as far back as [Mil88] and [CMR90, CMR91]). However, these simple measures do not capture the full richness of device differences, for example, one of the authors recalls Milner's presentation of his review of input device performance [Mil88], where he showed images of arcade gamers who made significant use of the way a trackball continues

to spin after being struck with the hand (sadly not reported in the paper itself!). Buxton also long argued for a richer view of devices with analysis of different kinds of children's drawing toys (such as Etch-a-Sketch) and using a simple finite state model to distinguish key difference between mouse, stylus and touch-based interaction [Bux86, Bux90]. Buxton's early work emphasised the way the lexical-level design of the physical interface can simplify syntax in interaction; a similar point is made by Usher et al. who refer to the 'digital syntax' embodied in the physical design of *token+constraint* tangible user interfaces [UIJ05].

More recently the importance of the physical nature of devices has re-emerged in several areas. In research in tangible user interfaces, the precise form of tangible tokens is usually designed taking into account the specific application and sensing technology [Ish08]. Also as user-interaction design has begun to overlap with product design the importance of physical form has become essential [BoV90].

2.3. Modelling physical and continuous action

Formal modelling in human-computer interaction dates back over 25 years (e.g. [Rei81, Suf82, DiR85, ThH90, PaP97]). Most is focused at the level of the dialogue starting at symbolic actions such as 'key A pressed' and on the behaviour of the digital system; although some work includes models of the physical systems being controlled by the digital device and even the user's mental states or behaviour so that conjoint properties can be investigated (e.g. [YGS89, CuR07]). Modelling of the physical aspects of interaction devices seems rare, a notable exception is Thimbleby's recent work modelling of the layout of controls [Thi07]. This work builds on long standing finite-state models of consumer devices, such as a VCR, and augments the FSM model of the digital behaviour with a specification of the precise physical location of buttons allowing detailed timings to be estimated for multiple button presses using Fitts' Law.

To date, there are only a few formal approaches to ubiquitous and tangible interaction. The ASUR framework [DSG02, DuG07a] focuses on the arrangement of devices, people and software components in the environment and the flows of data between them. ASUR has been applied to systems embedded into the environment as well as more self-contained devices including a fairly complex bespoke device, GE-Stick, for interacting with Google earth [DGR07b]. Building on roots in multi-modal systems, the Mixed Interaction Model also deals with the structure and flows between sensors and actuators within hybrid physical-digital devices [CoN06, CoN08], and has been applied to the design of a hand-held photo browser, not unlike that in Sect. 8 of this paper. Like ASUR, the approach is focused principally at the flows and relationships between sensors, but also includes tool support down to concrete implementation. At a more detailed level the uppaal system, which is based on timed automata, has been used to model a navigation system to guide visitors in an office building [HKC07]. While addressing a ubiquitous application, the features modelled in uppaal are principally those of the information system with the knowledge of the physical layout of the building embodied in a single black-box function 'sensorlink()' giving the closest navigation display to a user's location.

One of the crucial differences between the physical and digital worlds is *continuity* in terms of space, value and time. In this paper, we wish to represent simple phenomena as simply as possible, and so we will largely avoid modelling continuous activity and values. However, we shall find that the intrinsic continuity of physical world asserts itself, and issues such as in-between positions of switches, muscle pressure, and timed behaviour have to be considered.

To our knowledge, the earliest work that systematically addresses issues of continuity is status-event analysis. This includes formal models, specification notations and a conceptual vocabulary for dealing with such systems [Dix91b, DiA96]. Most recently this has been developed into an XML-based implementation notation XSED [DLF07]. Status-event analysis distinguishes event phenomena such as the (moment of the) pressing of a button, from status phenomena, such as the position of a mouse or the current temperature. Going further back, this distinction is implicit in the difference between event from sampling devices in GKS and very early formal device models [Ans92].

Most specification and modelling techniques for continuous interaction use two linked specifications largely separating the continuous and discrete parts. The discrete part is some form of discrete state system with event transitions whereas the continuous part defines status-status mappings that relate continuous input and output phenomena. The two are linked in that critical changes in status phenomena are treated as events (status-change events) and the exact form of the status-status mapping depends on the current state of the discrete system.

Two early examples of this are Interaction Object Graphs [Car94], which used a variant of Harel State charts for the discrete part augmented with data flows to capture the continuous interactions; and the PMIW user interface management system, which managed status-status mappings using a data-flow notation, which is effectively

‘rewired’ by a discrete finite state machine driven by event inputs [JDM99]. The former was used to model novel on-screen widgets and the latter to model interactions in virtual environments such as grabbing and moving objects.

Various forms of Petri Net have also been used to specify virtual environments [MDS99, WiH00]. In particular, Smith has recently used Flownets to model haptic interaction in virtual environments [Smi06]. As in Interaction Object Graphs and PMIW, Flownets use data flows to model the continuous (status–status) aspects of the system, with Petri Nets for the discrete aspects. The two are linked with the data flows being able to initiate tokens into the Petri Nets through threshold triggers (status-change events) and the presence of tokens at particular points being able to regulate the dataflow using ‘flow controls’.

An exception to the use of largely separate discrete and continuous components was Wüthrich’s work using cybernetic theory to model both discrete and continuous parts within what is effectively a purely continuous paradigm [Wüt99]. Within this paradigm discrete events become continuous functions that are zero/undefined except at the exact moment when the event occurs. This use of continuous techniques originating from a physics/engineering background is rare, however recent work by Eslambolchilar has used control theory to model human control of interface elements as a single (typically closed-loop feedback) system, which has been applied to a number of screen-based controls and mobile devices [E06].

3. Modelling approach

In Sects. 4–7 we will examine a number of properties of physical device interaction that have been identified from our previous studies of consumer products [GhD03, GhD05]:

- exposed states
- bounce back
- time dependent devices
- controlled state and compliant interaction.

This is not the complete list of properties uncovered in previous analysis, but includes those where the more formal analysis of this paper adds insight, or where the nature of the property requires additional elements in the physigrams. For example, another property is the ‘natural inverse’: whether physical movements that are naturally opposite (push/pull, etc.) cause opposite system changes. While the natural inverse is important and is currently being studied in detail, it is most centrally about the physical layout of controls and how these link to human motor movements, and so lies outside the scope of this paper. The natural inverse has interesting parallels with Task-Action Grammar [PaG86], which was specifically formulated in order to deal with natural relationships in textual interaction (e.g. a command ‘U’ for ‘up’ should be matched with ‘D’ for ‘down’). This suggests that adding a model of physical form could add significant value to physigrams; however, for this paper we focus on behaviour only.

Each of the sections will start with an informal example of consumer devices exhibiting the property, from a light switch to a washing machine. The example is then specified using an (augmented) STN of the physical device itself (the physigram) and an associated STN of the underlying logical state. As the sections unfold extra features are added to the physigram. Alongside the development of the physigrams themselves, each section also includes development of a more abstract formal model that directly encodes the elements in the physigram and logical state STN, and allows the specification of properties of them and their relationship. The formal model is itself grounded by showing the example systems described in the developing model.

The formal model gives what could be regarded as a surface semantics to the diagrammatic examples; that is it directly models the elements in the diagrams, but does not relate them to any independent semantic framework such as a physical model of the world. This is of course the normal level of semantic description in interface specification and indeed formal specification in general. However, the fact that we are focused on physical devices does suggest that some deeper semantics would also be of value, not for day-to-day use, but in order to give stronger foundations and in order to link different kinds of models.

The physigrams are basically slightly augmented STNs applied to the physical device. There were several reasons for using STNs rather than some other formalism.

First is simplicity. Only one of the authors is a formalist and in particular the author who performed the majority of the analysis of existing devices does not have any training or experience in formal notations or analysis. The physigrams have therefore been developed to support the needs of those without a formal background. The

comprehensibility of STNs is evidenced by the fact that they are used in end-user documentation; for example, an STN used in digital watch documentation is reproduced in the ‘dialog notations and design’ chapter of [DFA04]. Because STNs are relatively easy for non-experts to interpret, they are also used to communicate user interface issues to broad or popular audiences, for example, Degani uses STNs extensively in ‘Taming Hal’ [Deg04]. In fact, Degani includes an illustrative example where an STN is used to specify a light switch (p. 14) in a way that looks very close to the early examples of physigrams here, but does not distinguish the switch from the light it controls.

This does not mean that STNs are without problems, and we are aware that it is far more difficult to create STNs than to read them; in particular our experience using user interface STNs with students and at tutorials has been that novices find it difficult to distinguish activities/events from states. Interestingly this did not seem to be a problem with any of the physigrams produced by designers in Sect. 8; possibly this is because for a physical device the states are far more apparent than for an user interface where states may be hidden or the appropriate level of abstraction unclear.

As well as being relatively simple, STNs are actually quite powerful and variants of state transitions have been used for user interface specification from Parnas in 1969 [Pa69] to Thimbleby’s long term body of work looking at automated analysis of consumer user interfaces and his recent book ‘Press On’ [Thi07]. Indeed, many of the alternative formalisms are either variants of STNs or can be rewritten as STNs for practical examples. The most obvious alternative would be statecharts [Har87] as used in UML and used also for more complex examples in both Degani and Thimbleby’s books [Deg04, Thi07]. The parallel behaviour of the device STN and logical system STN could be described in a single statechart, but this did not seem to add additional expressivity beyond juxtaposing the STNs informally.

Finally, we use STNs because their simplicity means they embody fewer assumptions/biases than more complex notations, which, by their nature, tend to overcommit—especially when attempting to specify continuous behaviour in finite notations [DiA96b]. This will become evident in Sect. 7 when we discuss the way system events may change device states in a way that may ‘fight’ with user actions. If, for example, we had used statecharts to model the link between physical device and logical state, this would have included a default semantics for synchronisation, which would not have actually expressed the physical situation.

In general we have tried to avoid shoehorning the example devices into a pre-defined notation and instead attempt to flexibly change the notation to express the problems and features clearly and with verisimilitude. Having done this, it would be possible to look at each of the new features and ask how they would be modelled in, or be added to other notations, such as statecharts or timed Petri nets, although always doing so with an awareness of the intended audience of the resulting notation.

4. Exposed states and physical–logical mapping

4.1. Example: up/down light switch

One of the simplest examples of a physical device is a simple on/off light switch. In this case the switch has exactly two states (up and down) and pressing the switch changes the state (Fig. 2a).

Actually even this is not that simple, as the kind of press you give the switch depends on whether it is up and you want to press it down or down and you want to press it up. For most switches you will not even be aware of this difference because it is obvious which way to press the switch . . . it is obvious because the current state of the switch is immediately visible.

Note that the switch has a perceivable up/down state whether or not it is actually connected to a light and whether or not the light works.

The logical system being controlled by the device also has states and Fig. 2b shows these in the case of the light bulb: simply on or off. (In fact the light bulb may also be broken, but we will ignore faults in this paper.)

Of course in the case of a simple light switch, the states of the physical device are in a one-to-one mapping with those of the logical system being controlled. In previous work we have used the term *exposed state* [GhD03] to refer to the way that the perceivable state of the device becomes a surrogate for the logical state and makes it also immediately perceivable. In the case of turning on an incandescent light bulb in the same room as the light switch, this is a moot point as the semantic feedback itself is immediate and direct. However, in some cases there may be a delay in the semantic response (e.g. neon lights starting up, kettle when first turned on) or it may be hidden (e.g. external security lighting); in these cases the feedback inherent in the device is not just very obvious, but may be the only immediate feedback.

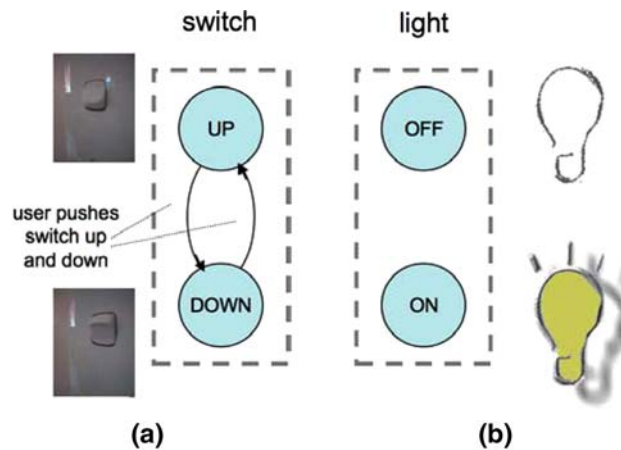


Fig. 2. Light switch: a physical device, b logical states

4.2. Formal model

We can model this kind of behaviour more generically. We denote by UA the set of potential user actions such as ‘push up’; these may be particular to a specific device ‘push button A’ as our environment affects our action possibilities. We use PA to denote the set of perceivable attributes of the world ‘light is shining’, ‘switch is up’. The full perceivable state of the world is composed of the different perceivable effects and there may be masking effects, e.g. if light 1 is on we may not be able to tell that light 2 is also on. However, for simplification we will just assume these are individually identifiable—at least potentially perceivable.

The physical device we model as a simple state transition network:

- DS – physical states of device
- $DT \subseteq DS \times DS$ – possible device transitions

In the light switch every transition (only two!) is possible, but in some situations this may not be the case. Hence the physically possible transitions are a subset of all conceivable from-to pairs. Note, for brevity, we will assume that the physical device states DS consist solely of those reachable through physically realisable transitions DT . For example, if the device consisted of two seamless hollow balls, we would not include the state where the smaller ball ‘magically’ was inside the larger.

Some of these transitions are controlled by user actions:

$$\text{action: } UA \leftrightarrow DT \text{ – n – m partial relation}$$

Note that this relation is n-to-m, that is the same user action may have an effect in several states (with different effect) and a single transition may be caused by several possible user actions (e.g. pressing light switch with left or right hand). In addition neither side is surjective, some physically possible transitions may not be directly controllable by the user (e.g. lifting large weight, pulling out a push-in switch) and some user actions may have no effect in the device in any state (e.g. blowing your nose). However, for exposed-state devices we will normally expect that the states are completely controllable by the user within the physical constraints of the device:

$$\text{controllable-state} \equiv \text{action is surjective}$$

Aspects of the user’s state may be perceivable by the user:

$$\text{ddisp: } DS \rightarrow PA$$

And in the case of exposed state each device state is uniquely identifiable by its visible or other perceivable attributes:

$$\text{exposed-device-state} \equiv \text{ddisp is injective}$$

Finally the logical system also has states which themselves may be perceivable via the feedback loops C or D.

LS — logical states of system
 $ldisp : LS \rightarrow PA$

For any system we can define a map describing which device states and logical states can occur together:

$state\text{-}mapping : DS \leftrightarrow LS$









The precise nature of this mapping depends on the operation of the system. In some cases like the light switch this is a one-to-one mapping between the physical device and logical states and this is precisely what we mean by *exposed state*.

$exposed\text{-}state \equiv state\text{-}mapping$ is one-to-one

This concept of exposed state is similar to some of the strongest forms of *observability* in the early formal methods in HCI literature [DiR85, Dix91]: the internal state of the digital system is completely observable through the external appearance or other physical properties of the device. However, exposed state is stronger than these observability properties as the means to manipulate the state are exactly those through which the state is observed. Looking back to this early literature, the system is also completely *predictable* so long as the user understands (a) the manipulations possible on the device and (b) the mapping between device states and system states.

4.2.1. Modelling the example

The mapping between the diagram components and the model is direct. As an example, we can express the light switch from Fig. 2 as follows:

DS = {UP, DOWN}
 DT = {< UP, DOWN >, < DOWN, UP >}
 UA = {PUSH_UP, PUSH_DOWN}
 $action$ = {PUSH_DOWN \mapsto < UP, DOWN >, PUSH_UP \mapsto < DOWN, UP >}
 PA = {, , , }
 $ddisp$ = {UP \mapsto , DOWN \mapsto }
 LS = {OFF, ON}
 $state\text{-}mapping$ = {UP \mapsto OFF, DOWN \mapsto ON}
 $ldisp$ = {OFF \mapsto , ON \mapsto }

Note that $ddisp$ is injective so it is an *exposed-device-state* device and also $state\text{-}mapping$ is one-to-one so the system has *exposed-state*.

4.3. Physical constraints as dialogue

The physical nature of a device puts limits on what can be achieved with it. We made DT a subset of conceivable transitions because some potential transitions between states may not be physically realisable; for example, a dial may not be able to move from setting 1–3 without first going through setting 2. Also we have discussed how there may be states that are conceivable, but cannot be achieved through reasonable physical transitions . . . at least not without physically breaking or dismantling the device.

In traditional UIMS and user interface architecture literature, one of the central concepts is dialogue—the component(s) responsible for the order and interpretation of interaction depending on context. However, physical interaction is usually placed at the lowest levels, separated from the dialogue by an intermediate layer (presentation in Seeheim [PfH85], lexical/logical interaction in Arch-Slinky [UIM92], and both presentation and interaction toolkit components in PAC-Amodeus [NCo91]). This is largely because the assumption underlying these architectures is that the physical devices are generic and do not have any constraints on their interaction—on a keyboard any key may be pressed at any time. With such an assumption, constraints on the user's possible interactions are governed by the way in which the unrestricted physical interactions of the user are interpreted by the system—dialogue imposed by software. In contrast more specialised devices may impose constraints on

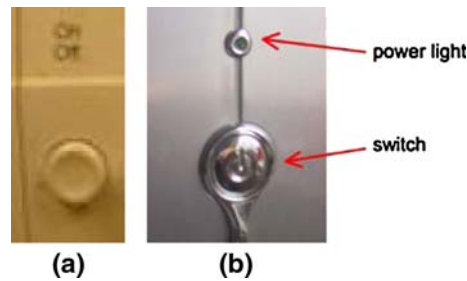


Fig. 3. **a** On/Off control with bounce back—is it on or off now? **b** On/Off button with indicator light

the possible interactions by virtue of their physical design; precisely the issue identified in Buxton's early work on simplifying dialogue syntax through physical design [Bux86, Bux90] and Usher et al.'s concept of 'digital syntax' [UIJ05]. As an example consider a mobile phone where the keyboard needs to slide open (which as a side effect causes a software event) before it can be seen and touched . . . and hence used—dialogue imposed by hardware.

In fact, discussion of this form of physical constraint can be found in some of the early work on formal modelling of user interfaces in the concept of a 'language' for the PIE model [Dix91]. The example used in this early work was a cash dispenser or ATM, as many ATMs at that time had a shield that covered the keypad and screen; only when a bank card was inserted was it possible to type at the keypad. These physical constraints can be seen as imposing a form of dialogue. For example, the old ATM effectively had a (low-level) dialogue of the form:

```
ATM ::= Transaction*
Transaction ::= CardIn [0-9]* CardOut
```

Note too that CardIn and CardOut must alternate—you cannot remove the bank card unless it is in the machine! This constraint is even true of generic keys and mouse buttons where 'press' and 'release' events must alternate. Physical interlocks are also common in machinery and industrial plant where the consequences of doing things out of proper order can be catastrophic.

This restriction of dialogue through constraints on possible interactions is not only found in 'real' physical interfaces. Even in WIMP interfaces a dialogue box, menu or window must be displayed on screen in order for the user to interact with it. It is often not necessary to have a software rule that says 'user is not allowed to press button A unless condition C holds' as one simply makes sure that button A is not visible on screen. This embedding of dialogue into the virtual 'physical' constraint of what is available on-screen, is probably the reason for the low emphasis on *explicit* dialogue management in much GUI interface construction.

For the designer of digital devices, the embedding of what would otherwise have to be software dialogue into physical constraints, can be an opportunity to both reduce the complexity of the digital interaction and make the interaction more intuitive.

5. Bounce back buttons

5.1. Example: push on/off switch

A more complex behaviour occurs with bounce-back buttons or other devices where there is some form of unstable state (pressed in, twisted) and where you need to exert continuous pressure in order to maintain the state. Figure 3a shows a typical example of a computer on/off switch. One press and release turns it on, a second turns it off.

Note that the user action here, pressing the button, is not discrete, but involves pressing until it 'gives' then *releasing*. While you maintain pressure the button stays in, it is when you release that it comes out again. However, the button comes out not because you pull it out with your release of pressure, but because it is internally sprung—the bounce-back.

Bounce-back buttons are found everywhere, from keyboards, to mice, to television sets. They typically do not expose the state of the underlying system as there is just one stable state of the physical device and it is the history of dynamic interactions with it that is important (on or off, what channel). The temporary unstable states of the

device are distinguishable, so long as pressure is maintained, but the device returns to its stable state as soon as the pressure is released. That is, the physical device itself does not maintain a record of its interaction in the way a rocker switch does.

Because the device does not itself expose the state of the underlying system (there is no feedback loop A for the state) we get potential problems of *hidden state* [GhD03]. Sometimes this is not an issue because the semantic feedback (loop C or D) is sufficient—for example, switching channels on a television set. However, even where there is semantic feedback this may be ambiguous (switching channels during an advertisement break) or delayed (the period while a computer starts to boot, but is not yet showing things on screen). In such cases, a supplemental feedback (loop B) close to the device is often used, such as a power light on or near the switch (as in Fig 3b).

Where the device does not have any intrinsic perceptible tactile or audible feedback (e.g. click of the switch or feeling of ‘give’ as it goes in) then supplemental loop (B) feedback may be given for the transitions as well as the states. Simulated key clicks or other sounds are common, but also, more occasionally, simulated tactile feedback can be used, as in the BMW iDrive.

From a design consideration, indirect feedback, whilst less effective, is useful in several situations:

- Where the complexity of the underlying system exceeds the potential states of the device, a simple one-to-one mapping is not possible. For example, a channel dial could work in a one-to-one mode if there are half-a-dozen channels, but not if there are hundreds of channels to choose from.
- Where we want to generate something that is perceived of as an event or action on the system. For example, the big red button (beloved of B movies) that fires a missile; here a rocker switch would make no sense, you can’t ‘unfire’ the missile by pushing the switch back.
- Some logical state transitions may be under internal system control. For example, a computer may be turned on using a push button, but switching off may be ‘soft’, under the control of the computer, to prevent accidental data loss. In Sect. 7 we will return to this issue and see how exposed-state can be consistent with system control. It should be noted that neither of the computer on/off switches photographed in Fig. 3 are on computers which can power themselves down in this way.
- The fact that continuous pressure is required can be used explicitly in tension states [Dix91] in order to manage temporary modes. Modes in interfaces have long been known to be a problem; the meaning of an action depends on the mode and so the effect of an action may not be as intended if the user does not know the current mode. Additional feedback is often added to make modes perceptually obvious most frequently visually (e.g. cursor shape), but also aurally [Mon86]. Associating modes with tension states mean that there is *haptic* feedback so it is hard to ‘forget’ that you are in a mode. For example, moving the mouse with a button pressed might draw a line rather than simply move the cursor, but users do not confuse the two as they can feel their finger pressing the button.
- A special case of tension state modes is when there is some sort of time-dependent effect in a mode (e.g. velocity-base joysticks). We will discuss time-dependent devices in Sect. 6.

5.2. Formal model

We can inherit much of the same formal machinery developed for simple exposed-state devices. However, in addition to transitions controlled by the user, we have bounce-back transitions. We label them Z (after Zebedee and the fact that Z and S are used for left- and right-handed helices). In the example here there is only one action leading to the states and thus it is clear what kind of user tension needs to be released in order for the bounce-back to happen. Sometimes (and we will see an example later) there is more than one tension simultaneously for a state (e.g. twist and pull), so we need to label the bounce-backs by the kind of release of tension (in terms of user action) that is being released.

$$Z : UA \leftrightarrow DT$$

The states that are the subject of bounce-back transitions are transitory states for that user action:

$$\forall a \in UA \text{ transitory-states}(a) \equiv \{d \in DS \text{ st. } \exists (d, d') \in Z(a)\}$$

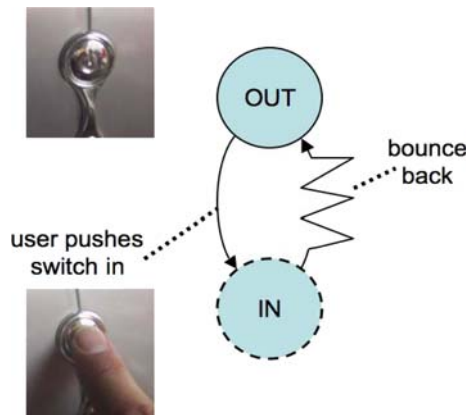


Fig. 4. States of bounce-back button

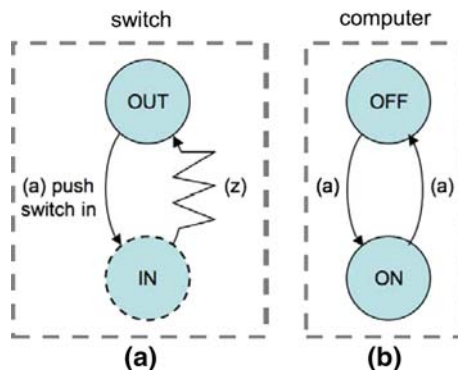


Fig. 5. a Physical states changes trigger event (a), b logical state changes based on events

Furthermore a transitory state for a user action cannot be the source of the same user-controlled transition and must have been reached by that user action:

$$\forall a \in UA \text{ transitory-states}(a) \cap \text{dom}(\text{action}(a)) = \{\} \\ \wedge \text{transitory-states}(a) \subseteq \text{range}(\text{action}(a))$$

Figure 4 shows the example of the computer switch with the bounce-back transition shown as a zig-zag line (spring) and the transitory state (IN) dotted.

While exposed state devices can have a one-to-one mapping between logical states and physical states, here the relationship is based on the events. Formally we define this first by associating events from a set Ev with physical state transitions:

$$\text{trigger} : DT \rightarrow Ev$$

This mapping may be partial as not every transition will cause an event. Also it is typically the case that only user-controlled transitions cause events ($\text{dom}(\text{trigger}) \subseteq \text{range}(\text{action})$), because once you have pressed a switch you are committed. However, there are exceptions such as the ‘drop’ (release the button) when you drag and drop with a mouse.

These events then cause transitions in the logical system:

$$\text{doit} : Ev \times LS \rightarrow LS$$

Figure 5 shows the physical device STN annotated with an event (a) and the effect of the event on the logical state (computer power). Note that in this example (and it is common!) there is no reason why the system could not have been designed with exposed state, for example a button that stays depressed and requires an extra push to release it. This design choice is often motivated by the aim to have a smooth surface although in the example

in Fig. 3b the switch is part of an embellishment anyway, so even this aesthetic reason seems to be absent. Finally, one of the reasons listed for having bounce-back is when the system is able to power itself down (or some similar logical state transformation), however, as noted previously, this is not the case for the switch in Fig. 3b.

5.2.1. Modelling the example

As in the previous section we can apply this to Figs. 4 and 5:

DS	$= \{OUT, IN\}$
DT	$= \{< OUT, IN >, < IN, OUT >\}$
UA	$= \{PRESS\}$
$action$	$= \{PRESS \mapsto < OUT, IN >\}$
Z	$= \{PRESS \mapsto < IN, OUT >\}$
$transitory-states(PRESS)$	$= \{IN\}$
PA	$= \{\text{img1}, \text{img2}, \dots\}$
$ddisp$	$= \{OUT \mapsto \text{img1}, IN \mapsto \text{img2}\}$
LS	$= \{OFF, ON\}$
Ev	$= \{(a)\}$
$trigger$	$= \{< OUT, IN > \mapsto (a)\}$
$doit$	$= \{< (a), OFF > \mapsto ON, < (a), ON > \mapsto OFF\}$
$state-mapping$	$= \{OUT \mapsto OFF, OUT \mapsto ON, IN \mapsto ON\}$

We can verify the conditions on *transitory-states*:

$$transitory-states(PRESS) \cap dom(action((PRESS))) = \{IN\} \cap \{OUT\} = \{\}$$

$$transitory-states(PRESS) = \{IN\} = range(action((PRESS)))$$

Looking at *ddisp*, it is injective, so, like the switch in Fig. 3, this too is an *exposed-device-state* device; the IN and (transitory) OUT states are distinguishable. However *state-mapping* is not one-to-one so the system does not have *exposed-state*.

5.3. Recapitulation: the exposed state switch

Using this expression of push-back we could in principle use this to model in greater detail the exposed state switch capturing the fact that pressure has to be initially exerted and some slight give is felt until the switch eventually yields and flips to the new state. Figure 6a shows this with transitory states for when the switch is up and just being pushed down. If you release before putting sufficient pressure on it snaps back to UP, but if the pressure is sufficient the switch yields and goes to the new state.

This yielding is rather like bounce back in that once the critical point is reached the device just goes of its own accord. However, we have drawn it slightly differently (less of a spring and more of a lightning bolt) in order to emphasise that this is going ‘with’ the user’s action and it is the point at which the ‘commitment’ occurs.

Note that in Fig. 6a a transition is included for ‘press up’ in the UP state which simply leaves the switch in the UP state. This distinguishes ‘press down’, for which there is a little give with a small pressure, from ‘press up’, for which there is no give. Thus we can begin to capture some of the nature of Gaver’s sequential affordances (described below).

In fact to model this completely we would need to include degrees of pressure and the fact that there is not just one half pressed-down state, but a whole series requiring increasing pressure. This is not captured by a finite state diagram or description and would require a full status–event description as we are talking here about *interstitial behaviour* (the interaction between events) and status–status mapping (more pressed = more down) [DiA96]. This is also reminiscent of Buxton’s three-state model for pointing devices, where the degree of finger pressure is one of the critical distinctions between abstract states [Bux90].

Figure 6a is rather complicated and, whilst useful in order to clarify detailed behaviour, would be a little noisy for real design use. Figure 6b shows a shorthand that emphasises the slight give of the press down action in the UP state by the comic-book-style movement arcs. In fact, we will often omit even this, as in many cases every action has this slight give property. However, in Sect. 8 we will see examples where explicitly marking give vs.

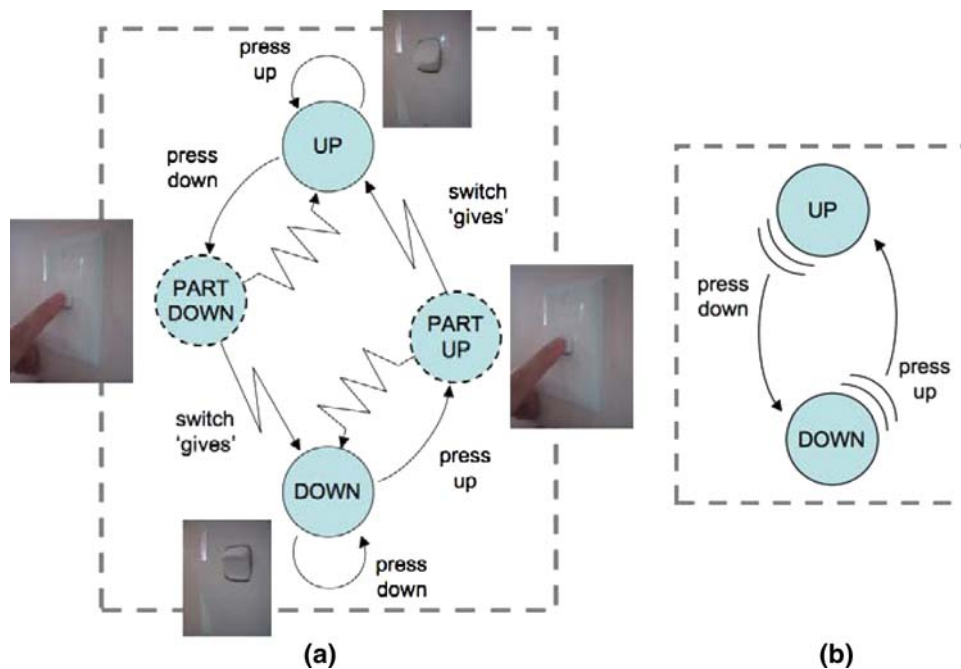


Fig. 6. Capturing initial pressure on exposed state switch: **a** detailed model using bounce-back, **b** more convenient shorthand

non-give transitions distinguishes otherwise similar physigrams. This form of shorthand would be also be useful in cases where some controls are operated on the slightest pressure—typically fully electronic ones. Formally we can capture this ‘give’ in some transitions by a simple ‘has-give’ predicate over user actions in particular states.

5.4. Give and sequential affordance

Note how even in this simple example of flipping a switch, the user actions are not simply ‘events’, but are protracted over time and vary in force. An additional way in which the user gets feedback on the state of the device and appropriate actions is by ‘trying out’ an action, often unconsciously, and if there is a small ‘give’ in the device continuing the action and increasing pressure until the user action causes a change in state of the device.

The importance of this effect is hinted at by Gaver when he introduced the notion of *sequential affordances* [Gav91]. Gaver discusses a door handle which initially just looks the size of your hand, so invites grabbing. However, once you have the door handle, it then has a second affordance, that of turning. It maybe that in that early paper Gaver simply meant (in the sense of Gibson’s affordances [Gib86]) that a door handle is not physically turnable by your hand until you have grasped it. However, it is also the case that when your hand is on the door handle you can feel a little ‘give’ and this tells you which direction to turn the handle, especially important when it turns in the ‘wrong’ direction.

This use of ‘give’ is clearly part of tacit design practice, for example, many cameras use a half-pressed shutter release button to mean ‘do auto focus’. However, to the authors knowledge, the issue is not discussed more explicitly in the HCI literature. This is perhaps surprising given the importance of the distinction between, say, a touch-based switch and one with a more solid button, but perhaps the lack of attention to such differences is simply due to the prevailing culture of device abstraction.

Indeed once the issue of ‘give’ is forefronted we can see digital equivalents, such as tooltips or the ability to slide off an on-screen button with activating it. This could also impact hardware design; for example, if mouse buttons had a ‘pressing but not fully-pressed’ state, rather like the camera shutter release, then this could be used as part of interaction to show some form of preview of the effect of the click, just like trying a door handle.

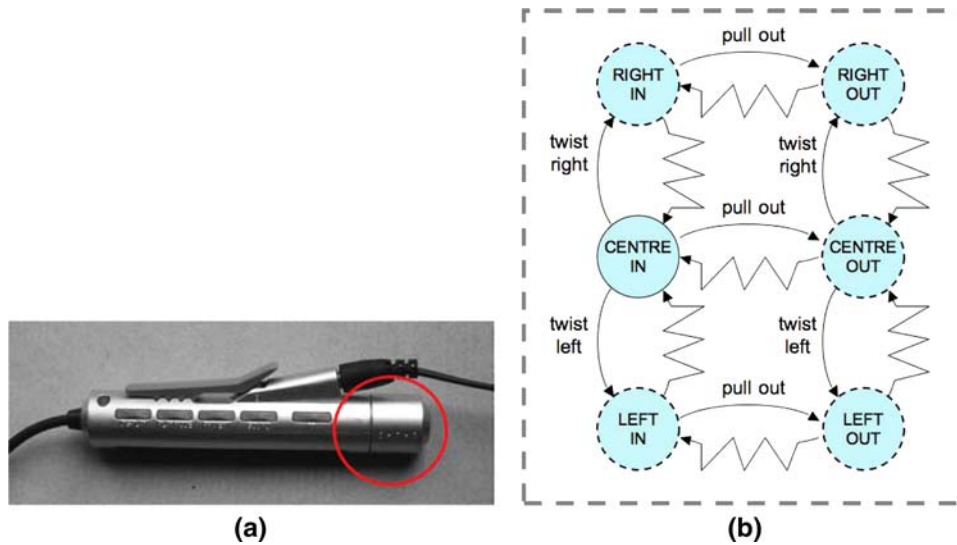


Fig. 7. **a** Minidisk controller; **b** device states

6. Time-dependent devices

6.1. Example: track/volume selector

Our next level of complexity includes devices, such as keyboards with auto-repeat or tuning controls on car radios, where things happen depending on how long you have been in a state. Figure 7a shows a minidisk controller. The knob at the end can be pulled in or out turned to the left or right. Figure 7b shows this physical state transition diagram of the device. This might be more succinctly described using two ‘concurrent’ STNs, one for in-out and one for left-right (as in state charts), but as they are coupled in a single control we are showing all the transitions to give an idea of the total complexity of even such a simple thing as one knob!

Whether the knob is pulled in or out determines whether it is affecting the volume or track selection and the amount of time it is pushed to the left or right moves the volume/track selection up or down. The former is a simple mode effect . . . and, as discussed in Sect. 5, a tension mode carries its own feedback, so is a good design feature. However, we shall focus on the left-right twists and their time behaviour.

To do this Fig. 8a shows just the left-right part of the diagram (actually in the ‘out’ condition) for when it is controlling track selection, and Fig. 8b shows the state diagram for the logical system, the selected track. As in Fig. 5 we use event labels to match the two. For this device we have had to augment the device transitions with additional timed transitions (labelled τ).

For some devices, there may be timed behaviour as part of the physical device itself, for example, eco-friendly light switches in hallways that slowly turn themselves off. However, for the minidisk controller, these timed transitions are not part of the physical device behaviour, but are strictly part of the device–logical state mapping; Fig. 8a is thus not the raw device STN. We have added them as *annotations* to the device STN, both for convenience, and also because the user is aware that the knob is *being held* for some time even if the exact times when events are triggered are not totally under the user’s control (unless they have a millisecond clock in their heads!). In Sect. 8.3 we will again see a need to ‘layer’ some additional information onto the raw device physigrams, but whenever we do this we need to be very careful as we are adding information that the designer knows, but may not be apparent to the user from the physical behaviour of the device.

From a usability point of view, these timed events have a special status as the user is not performing clear actions. In the case of the minidisk controller, the timed events are all in tension states increasing the user’s awareness that additional system events may occur. This follows one of the general design heuristics from status–event analysis that *trajectory dependent* effects (those where the path of movement matters, not just its end point) should normally take place only in tension states [Dix91]. A very easy ‘undo’ is even more critical for these implicit timed events than for more deliberate user’ actions. In the case of the minidisk controller there is no explicit ‘undo’,

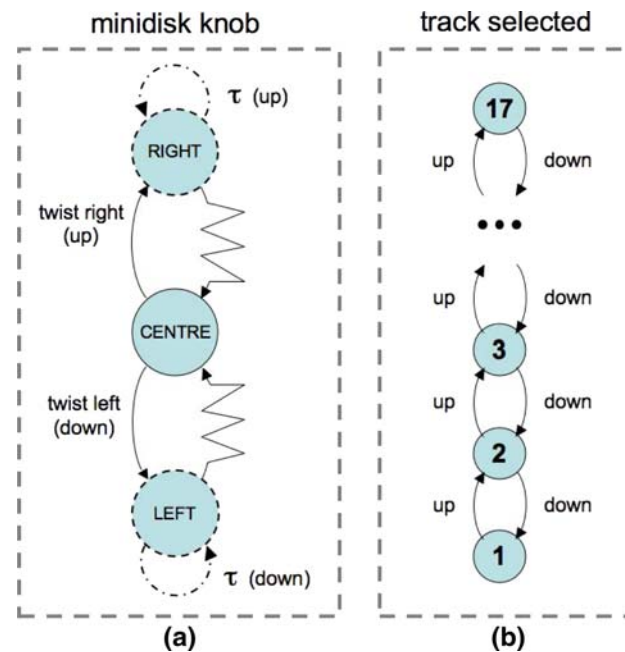


Fig. 8. Minidisk: **a** time augmented device **b** logical states

however simply turning the knob in the opposite direction creates the opposite effect. In fact, this form of *natural inverse* is a very easy way of allowing the user to effortlessly undo actions [GhD06].

In addition to timed events in or closely related to the device behaviour, there are often timing issues more deeply embedded in the digital behaviour; for example, many digital appliances revert to some default state after some period of inactivity. There is a long-standing literature on the importance of time in the user interface (e.g. see [Shn84] and chapter 5 of [Dix91]), but it is still often not given due attention in interaction design. However, given our focus on the physical device behaviour, we do not consider these more internal timings further.

6.2. Formal model

Notice that everything in Fig. 7b, with the exception of CENTRE-IN, is a tension state. However, there are actually two kinds of tension demonstrating why we needed to label transitory states and bounce-backs by user actions in Sect. 4.2.

In Fig. 8a we draw the timed events as if they were transitions, however we model them simply as an aspect of the state. This is because for the user the system does not make little transitions to the same state, it simply stays in the tension state. For the user the transitions happen *while* in a state not at some hidden transition in an invisible state model. This emphasises the importance of allowing the phenomena to shape the notation rather than fitting phenomena to the notation as discussed in Sect. 3. There may also be real timed transitions, but these are more often in response to things happening in the logical state, which we discuss in the next section. So all we need to do is say in which state and how frequently the timed events occur.

$$\text{time-trigger} : DS \times \text{Time} \times \text{Kind} \rightarrow Ev$$

Here *Time* is as in ‘gap between moments’ rather than time on the clock, and *Kind* is either *PERIODIC* or *SINGLE*. This is not totally general, but seems to capture most timed events seen in practice except complex continuous time effects such as mouse ‘acceleration’ settings. Note that in other circumstances we would be able to dispense with the special *SINGLE* case by adding an extra state. However, the states of the STN correspond exactly to the physical states of the device, so we cannot simply duplicate them corresponding to hidden electronic or digital transitions. The only situation where it would be appropriate to add time-based states to the device

STN would be when this is apparent in the state of the actual physical device, for example, in the way that some corridor light switches turn themselves off after a fixed time.

In terms of status–event analysis, these timed events are another example of interstitial behaviour. This again shows that a more fine-grained model would need to use a full status–event description and we would need to use some form of a real-time model to express precisely the detailed semantics of *time-trigger*.

6.2.1. Modelling the example

The physigram for the full minidisk knob in Fig. 7 can be modelled as follows:

$$\begin{aligned}
 DS &= \{\text{CENTRE_IN, LEFT_IN, RIGHT_IN, CENTRE_OUT, LEFT_OUT, RIGHT_OUT}\} \\
 DT &= \{\langle \text{CENTRE_IN, LEFT_IN} \rangle, \langle \text{CENTRE_IN, RIGHT_IN} \rangle, \langle \text{LEFT_IN, LEFT_OUT} \rangle, \dots\} \\
 UA &= \{\text{TWIST-LEFT, TWIST-RIGHT, PULL-OUT}\} \\
 \text{action} &= \{\text{TWIST-LEFT} \mapsto \langle \text{CENTER_IN, LEFT_IN} \rangle, \text{TWIST-RIGHT} \mapsto \langle \text{CENTER_IN, RIGHT_IN} \rangle, \\
 &\quad \text{TWIST-LEFT} \mapsto \langle \text{CENTER_OUT, LEFT_OUT} \rangle, \text{TWIST-RIGHT} \mapsto \langle \text{CENTER_OUT, RIGHT_OUT} \rangle, \\
 &\quad \text{PULL-OUT} \mapsto \langle \text{LEFT_IN, LEFT_OUT} \rangle, \text{PULL-OUT} \mapsto \langle \text{CENTER_IN, CENTER_OUT} \rangle, \\
 &\quad \text{PULL-OUT} \mapsto \langle \text{RIGHT_IN, RIGHT_OUT} \rangle\}
 \end{aligned}$$

$$Z = \{\text{TWIST-LEFT} \mapsto \langle \text{LEFT_OUT, CENTER_OUT} \rangle, \dots, \text{PULL-OUT} \mapsto \langle \text{LEFT_OUT, LEFT_IN} \rangle, \dots\}$$

$$\begin{aligned}
 \text{transitory-states} &= \{\text{TWIST-LEFT} \mapsto \text{LEFT_IN, TWIST-LEFT} \mapsto \text{LEFT_OUT,} \\
 &\quad \text{TWIST-RIGHT} \mapsto \text{RIGHT_IN} \dots, \text{PULL-OUT} \mapsto \text{LEFT_OUT}, \dots\}
 \end{aligned}$$

Note how the *transitory-states* include several types of user action for some states. For example, LEFT_OUT requires pressure of both TWIST-LEFT and PULL-OUT. Note also how *Z* records which state LEFT_OUT will drop back into when a particular pressure is released.

Moving on to the device–logical state mapping with its timed events, we will just consider the case when the minidisk knob is pulled out, as in Fig. 8:

$$\begin{aligned}
 LS &= \{1, 2, 3, \dots\} \\
 Ev &= \{\text{up, down}\} \\
 \text{trigger} &= \{\langle \text{CENTRE_OUT, LEFT_OUT} \rangle \mapsto \text{down}, \langle \text{CENTRE_OUT, RIGHT_OUT} \rangle \mapsto \text{up}\} \\
 \text{time-trigger} &= \{\langle \text{LEFT_OUT, 1sec, PERIODIC} \rangle \mapsto \text{down}, \langle \text{RIGHT_OUT, 1sec, PERIODIC} \rangle \mapsto \text{up}\} \\
 \text{doit} &= \{\langle \text{up, 1} \rangle \mapsto 2, \langle \text{up, 2} \rangle \mapsto 3, \dots, \langle \text{up, 16} \rangle \mapsto 17, \langle \text{down, 17} \rangle \mapsto 16, \dots\}
 \end{aligned}$$

7. Controlled state and compliant interaction

7.1. Example: washing machine and electric kettle

Finally we come to devices where the state of the physical device is affected by the underlying logical system as well as vice versa. Consider a washing machine control knob that sets the programme (Fig. 9a) or an electric kettle switch (Fig. 9b). In each case the user can control the device: twisting the knob to set the programme or pushing up or down the kettle switch to turn the kettle on and off. However, in addition the underlying logical system can also control the physical device. In the case of the washing machine as the clothes are washed the dial usually moves round to act as a display of the current state of the programme. In the case of the kettle, when the water boils many kettles both switch themselves off and at the same time release the switch.

We say that this kind of device has *controlled state*; that is the state of the physical device is not simply manipulated as in input by the user, but is also controlled as an output by the underlying logical system.

In fact both systems in addition exhibit *compliant interaction* [GhD03] where the system control of the physical device operates in a compatible way to the user control: with the kettle the user can turn the switch off or the system can and the effect on the switch is the same for both user or system control. Of course there are usually limits to compliant interaction: the kettle does not turn itself on and the user turning the knob to the end of the wash cycle does not magically wash the clothes!

Figure 10 shows the state diagram for the kettle switch and also the state of the power and water. Strictly there are two sub-systems in the kettle: the *power* (ON/OFF) influencing the *water temperature* (continuous

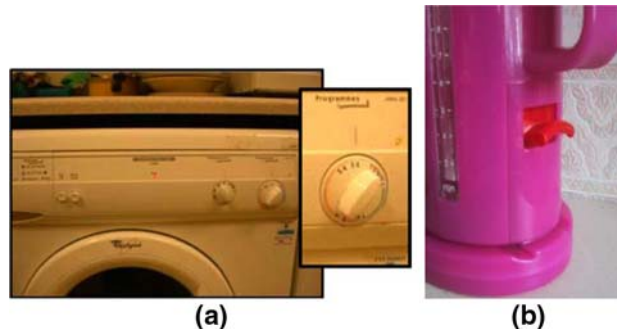


Fig. 9. Compliant interaction: a washing machine knob, b kettle switch

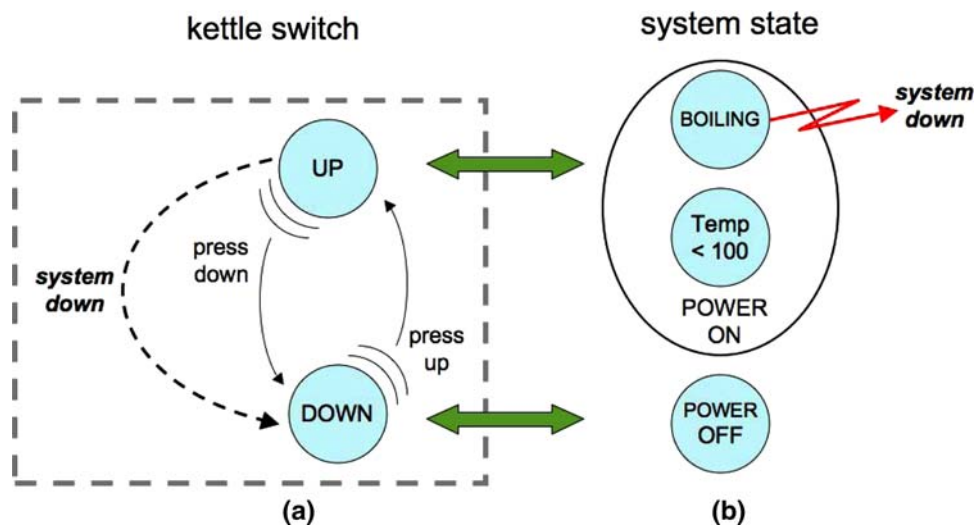


Fig. 10. Electric kettle: a kettle switch, b power and water

scale), but for simplicity we have shown the water state as simply boiling vs. not boiling and only as sub-states of the POWER-ON state. The arrows between the device and logical state show that there is an exposed state for the electrical power system. The little lightning arrow from the water’s BOILING state shows that simply being in the state, by itself, triggers the system action ‘system down’. Like user actions in the physical world this is protracted and lasts as long as the kettle is boiling, it is not simply an event at the moment boiling is first sensed. This possibility of an autonomous action is shown by the dashed transition on the state diagram for the physical switch.

Note how the system action and the user action to switch off the kettle are both operating in exactly the same way on the physical device. Note also that if the user is pushing up when the system is trying to switch the kettle off there is a conflict and whether the switch goes off or not depends on who is stronger! For most electric kettles the automatic switching off is usually weaker than the user’s ability to hold the switch up (usually simply releasing a catch) so it is possible to boil the kettle when dry. In some kettle designs the power is switched off by the system when the water is boiling irrespective of whether the user allows the switch to go down; in this case we would have similar device states, but different logical state transitions and no exposed state mapping.

Again note that if we used a notation with an in-built model of synchronisation between components, this conflict and alternative designs might be at best missed and at worst mis-specified. This is not to argue that one should not use more elaborate and notations, but that for this investigative analysis the simpler notation forces us to face important design issues.

7.2. Formal model

To deal with these kinds of devices we need to add a set of system actions SA and have a mapping that says which system actions are triggered by which logical states:

$$sys-trigger : LS \rightarrow set(SA)$$

These system actions will then have their effect on device state transitions just like user actions:

$$sys-action : SA \leftrightarrow DT \text{ --n-- m partial relation}$$

Just like user actions it is possible that a single system action may have different effects in different device states and that several system actions might be possible in a single device state. However, when it is an exposed state system, like the kettle, it is likely that the system actions are very specific for a particular state. Indeed if there is a state mapping, then there should be some consistency between the system state(s) that correspond to a device state and the system actions pertaining to each:

$$\begin{aligned} \forall s \in LS \quad \forall a \in sys-trigger(s) \\ \quad \exists d \in dom(sys-action(a)) \text{ st. } (d, s) \in state_mapping \\ \forall a \in SA \quad \forall d \in dom(sys-action(a)) \\ \quad \exists s \in sys-trigger^{-1} \text{ st. } (d, s) \in state_mapping \end{aligned}$$

Or equivalently:

$$\begin{aligned} \forall s \in LS \quad \forall a \in sys-trigger(s) \\ \quad dom(sys-action(a)) \cap state_mapping^{-1} \neq \emptyset \\ \forall a \in SA \quad \forall d \in dom(sys-action(a)) \\ \quad dom(sys-trigger^{-1}) \cap state_mapping \neq \emptyset \end{aligned}$$

In each case, the first of these says that if a logical state can trigger a system action then at least one of the device states consistent with that logical state must take account of that system action. The second says the converse, that if a device state can be affected by a system action then it must be possible for one of the logical states consistent with that device state to generate the action.

Either of these conditions may be broken, but that would suggest that some aspect of the physical device is not being fully utilised, or some signal from the logical device is being ignored. This may be an intended effect of the combination, but certainly merits checking.

7.2.1. Modelling the example

The kettle in Fig. 10 can now be modelled:

$$\begin{aligned} DS &= \{UP, DOWN\} \\ DT &= \{< verb + UP+, DOWN >, < DOWN, verb + UP+ >\} \\ UA &= \{PRESS-DOWN, PULL-UP\} \\ action &= \{PRESS-DOWN \mapsto < UP, DOWN >, PULL-UP \mapsto < DOWN, UP >\} \\ Z &= \{\} \end{aligned}$$

For this example, the logical system state itself is more complex. There are two sub-systems, power and water, which we represent by abstraction functions:

$$\begin{aligned} power : LS \rightarrow PowerState \\ water : LS \rightarrow WaterState \\ PowerState &= \{POWER_OFF, POWER_ON\} \\ WaterState &= \{NOT_BOILING, BOILING\} \end{aligned}$$

When, as in this system, the sub-systems are orthogonal (any combination of sub-system states is possible) and between them completely define the logical state, then LS is simply the Cartesian product of the sub-system states ($LS = PowerState \times WaterState$) and the abstraction functions are simply the component mappings.

Given such sub-system mappings we can define what it means for the system to exhibit exposed state relative to a sub-system:

$$exposed-state \text{ wrt. } power \equiv (power \circ state_mapping) \text{ is one-to-one}$$

This would be exactly the case for the kettle if the kettle is one of the simpler kind that allows you to hold the switch down to keep electricity on when the water is already boiling (it is at this point we can model some of the design alternatives):

$$power \circ state\text{-mapping} = \{DOWN \mapsto POWER_OFF, UP \mapsto POWER_ON\}$$

Finally we model the system actions:

$$\begin{aligned} SA &= \{\text{system-down}\} \\ sys\text{-trigger} &= \{< POWER_ON, BOILING > \mapsto \{\text{system-down}\}\} \\ sys\text{-action} &= \{\text{system-down} \mapsto < UP, DOWN >\} \end{aligned}$$

8. Physigrams in use

8.1. Context

Two of the authors are product designers. They are part of the Programme for Advanced Interactive Prototype Research (PAIPR), a research group attempting to create a suite of systems for the development of computer embedded products sympathetic to the designer's mindset and methods. There are a number of other groups working in this area, e.g. Phidgets [GrF01, Phi08], Voodoo Dolls [PSP99], DTools [HKB05], Switcharoo [AvH02], Denim [LaM95]. Unlike the work of these groups PAIPR has a product design focus rather than an electronics or programming base. PAIPR's methods centre around a system of working that involves low-tech keyboard emulation boxes called IE Units wedded to software building blocks [GLH05]. The system allows rapid prototyping without the usually requisite electronics or programming skills. The system has been used to empirically measure the performance of real products against physical and virtual prototypes and this research found that the link between the physical act of holding a product and interaction was more marked than has previously been understood [EvG06]. This has led to the group to become more interested in the precise nature of physicality in the design process hence the work with the physigrams.

The designers were not involved in the development of the work described in previous sections, in particular, they had not previously been exposed to the physical device STNs (physigrams). For the rest of this section we will refer to them as 'the designers' and in contrast describe the authors who were involved in developing the notation as 'the developers'. Both designers and developers are involved in a broader project on understanding physicality in design. So, there is some danger that the designers share more conceptual background with the developers than would be the case with a typical product designer. However for the exploratory purposes of this case study of use, we believe that the fact that the designers were previously not exposed to physigrams is sufficient to ensure valid results.

As part of a project meeting, the designers were first given a short explanation of the concept by the developers (approximately 10–20 min), in particular the developers introduced the notion of studying the physical device 'unplugged' from its digital aspects and some of the diagrammatic examples. The designers were then given an earlier paper [DGR07] that covers largely the same ground as Sects. 2–7. As the designers were not from a computing or mathematical background they were instructed to ignore the parts on the formal specifications, but to read those regarding the formal diagrammatic notations.

Subsequently, and without any further input or aid from the developers, the designers then read the relevant parts of the paper and spent a collaborative session applying the physigrams to an ongoing design project. It should be noted that the designers were not given a brief by the developers, but applied the techniques to a brief and project that they were pursuing for other reasons; that is the physigrams are effectively being used 'in the wild'. Because of this, the developers were not able to tune the brief to exercise all aspects of the physigrams, for example time effects. However, this free use by the designers has the advantage of not being limited to the developers' pre-conceptions; this gave the designers the freedom to explore issues that the developers may not have considered.

The existing brief the designers were working on involved three alternative devices for interacting with the same underlying application; this was in fact a good exercise for the physigrams as we were able to see how superficially similar, but subtly different devices were distinguishable in the physigrams. The designers were not given any time limit for using the physigrams, but in the end spent around an hour in total during which time they produced initial handsketched physigrams followed by two electronic variants of each of three design alternatives. Of this hour about half the time was spent in discussion and the other half producing the final physigrams.

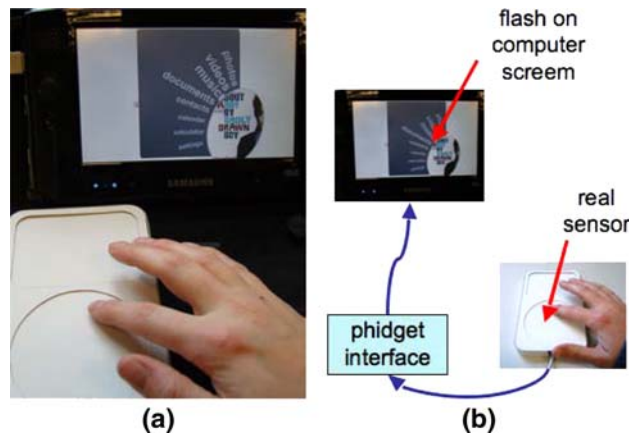


Fig. 11. Prototyping device



Fig. 12. Variants of prototyping device: a device 1, b device 2 and 3, c device 4

8.2. Prototypes

The design exercise they chose was one connected to the shared project involving creating multiple high-fidelity prototypes of an iPod-like device. Figure 11a shows one of the devices in action. The device has a round touch-sensitive area, rather like a circular trackpad, and also above it a rectangular area where the display would be. For the prototype the display is instead emulated in Flash on a separate computer screen, which shows the part-circular menu envisaged for the device.

The prototype is interactive as there is a real touch sensor inside the cardboard mock-up. The sensor is a Phidget [GrF01, Phi08] and the hardware and libraries supplied convert the raw sensor inputs into Flash events. Figure 11b shows the setup.

In all there are four versions of the prototype device, all with identical display functionality, differing only in the input device. Figure 12 shows the devices. Device 1 has a clicking rotary switch with a knob connected to an IE unit [GLH05] that can be turned to 12 different directions. Device 2 has a clicking rotary switch identical to device 1 connected to an IE unit, but it is turned using a flat dial rather than a knob. Note that the knob has a direction that can be detected visually and by touch. In contrast device 2 does not have any distinguishable direction. Device 3 is identical in appearance to device 2 (hence a single photo), but inside has a different rotary switch that rotates freely, but detects 12 orientations, again attached to an IE unit. Finally, device 4 is the one introduced in Fig. 11 with a touchpad.

All the devices also can be pressed to act as a ‘select’ event. In devices 1, 2 and 3 the whole knob or dial is depressed. In device 4 this is achieved by touching in the centre (NB. this had not been implemented in the prototype but for the purpose of the physigrams the design intent was to have the touch select in the middle).

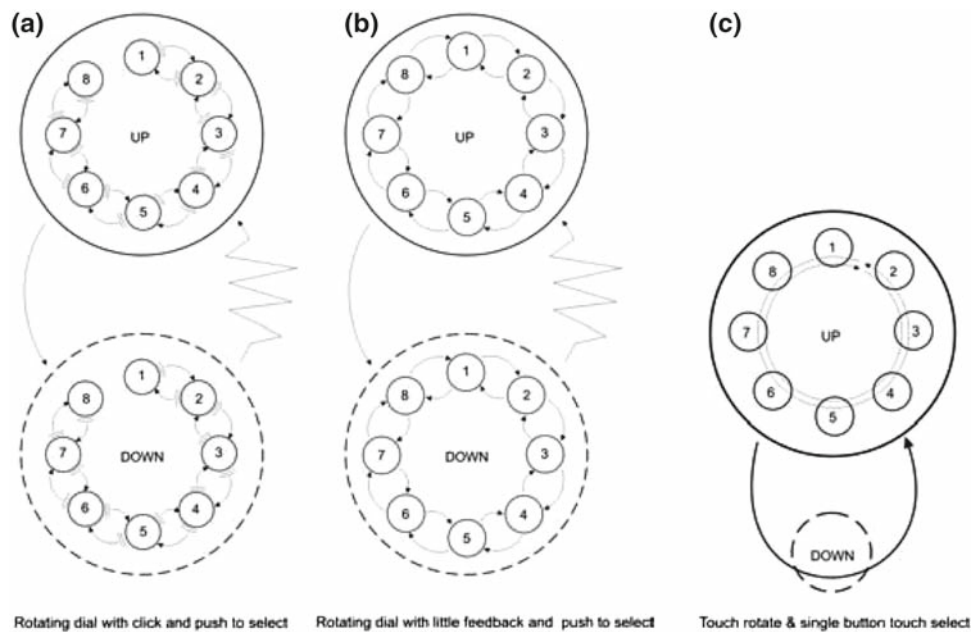


Fig. 13. Physigrams for each device: a device 1 and 2, b device 3, c device 4

8.3. Physigrams

Figure 13 shows the physigrams produced by the designers for each device. Devices 1 and 2 look different and have a visually different control, but have the same rotary switch connected to an IE unit inside and hence the same physigram (Fig. 13a). The physigram shows 16 states in total, 8 when the device is ‘up’ and 8 when it is ‘down’. As the press down on the device is sprung there is a bounce back between the down states and the up states. The designers took some liberty with the notation here and used the circle drawn round the 8 up states to bind them into a single ‘super state’. The implied semantics is that when you press up or down you end up in the same numbered state.

Note that there are only 8 states drawn yet there are 12 outputs of the IE unit. When discussing these, at first the developers thought this was a mistake and the designers had not fully grasped the ‘unplugged’ concept. If you turned the device 12 steps could clearly be felt, the 8 seemed to refer to logical states of the application. Indeed, the designers explained that initially they had drawn the diagram with 8 states labelled by the system functions they selected. However, they then realised that this was not an unplugged device. They considered drawing in 12 states each in the up and down circles, but in the end decided not to. This would have been an accurate description of the rotary switch used in the *prototype*—however, this was just a prototype and the intention was that if the device were actually produced an 8 state dial would have been used. That is they used the physigrams to specify the intended design not the limitations of the prototype.

Device 3 (Fig. 13b) is quite similar in broad terms. The most obvious difference is that there is no state 1 to state 8 transition for devices 1 and 2 as the dials do not rotate a full 360 degrees, whereas device 3 can rotate totally freely. Device 4 differs more radically still. Both device 3 and device 4 have free movement, however the difference is that whilst with device 3 (the freely tuning dial) it is possible to turn the dial whilst it is pressed in, in the case of device 4 (the touchpad) the press has to be in the middle, so there is no rotary movement in the ‘down’ state.

In fact, the physigram for device 3 also differs from devices 1 and 2 at the level of movement between the sub-states. This is evident in the close-up details in Fig. 14. Devices 1 and 2 have definite ‘stops’. As you try to turn the knob or dial there is slight resistance and then when you twist sufficiently the knob/dials clicks to the next position. That is, it is a bounce-back/give behaviour and the designers used the shorthand introduced in Sect. 5.3.

Looking at the detail for device 4 (Fig. 14c) recall that the user’s finger can move freely over the trackpad. The designers indicated this by the continuous circles, showing the finger can move freely in either direction. However,

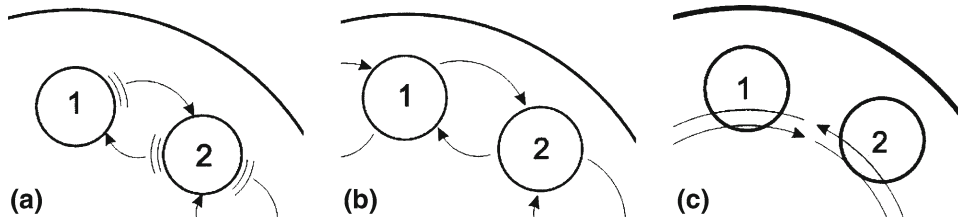


Fig. 14. Detail from physigrams for each device: **a** detail device 1 and 2, **b** detail device 3, **c** detail device 4

some locations are sensed by the system and used to represent states. The designers drew numbered states on this overlaying the continuous circles to show that different areas had different meanings. Here they again seemed to be not taking on board the ‘unplugged’ concept. However, they explained that they had considered this carefully and had chosen not to label the states with application names (as they had decided for devices 1 and 2), but they also knew that in the hardware the device reported only 8 states and so they used the diagram to represent this. This highlighted an ambiguity in unplugged-ness. There are actually three levels:

1. the physical device fully connected with its underlying application
2. the device with its internal electronics, but unplugged from its application
3. the purely physical aspects of the device

The physigrams were originally intended for level 1, but the designers had used them for level 2. Both are of course important. Indeed, this is exactly the same kind of issues that we found when considering timed transitions in Sect. 6. There the τ annotation in Fig. 8a referred precisely to level 2. This may mean that for certain devices we should consider drawing both level 2 and level 3 diagrams or perhaps annotating a single diagram to make clear which elements are level 2.

A problem with the use of level 2 physigrams, which we noted in Sect. 6, is that they embody knowledge that is available to the designers, but may not be apparent to users. For the timed transitions this was simply annotations to states, but here there are effectively states drawn that are not part of the physical device behaviour. However, insisting on level 1 diagrams would be counter productive as clearly describing level 2 features are of value also. Many applications used by practicing designers use notions of layers (e.g. Photoshop); so, one could envisage tool support that allowed multiple layers of annotations to be added to a basic physigram thus encouraging designers to consider both the raw device behaviour and also the way this interacts with low-level digital features.

From the description it would appear that device 3 and device 4 should have similar physigrams for the rotation part as both have totally free movement to any orientation. In fact the detailed view of device 3 looks more similar to devices 1 and 2. The designers explained that although the dial did not have click stops in the way that device 2 did, in fact it was just possible to feel when the dial moved past one of the contacts. That is while device 2 had haptic feedback (actual resistance) device 3 has tactile feedback (felt transitions). The designers had represented this by using a simple state change diagram for device 3 compared to the bounce-back in device 2 (similar state transitions, but different arrow shape). This makes a clear distinction between the two, but is not entirely satisfactory as it does not show the continuity of movement possible in the way that the physigrams for device 4 does. This is another example, as we have found previously in this paper, of the importance of being able to deal adequately with continuous phenomena.

8.4. Using the page

The formal meaning of the physigrams depends only on the topology and connectivity of states and arrows and not their precise positions. This meant that the designers were free to use spatial layout on the page to convey additional meaning. In some cases this represented aspects that one might want to capture formally, in particular the idea that the ‘big’ up/down transitions in Fig. 13a, b take you to the corresponding numbered sub-states. More often the page is used to help the human reader make sense of the diagram, making the form on the paper correspond roughly to the form of the device, as in the circular dial.

The designers in fact went through several iterations before ending up with the neat versions in Fig. 13, some of which was about interpretation of the more formal or semantic issues (such as whether to label the states abstractly 1–8 or by application labels), but much was also about the most useful layout.

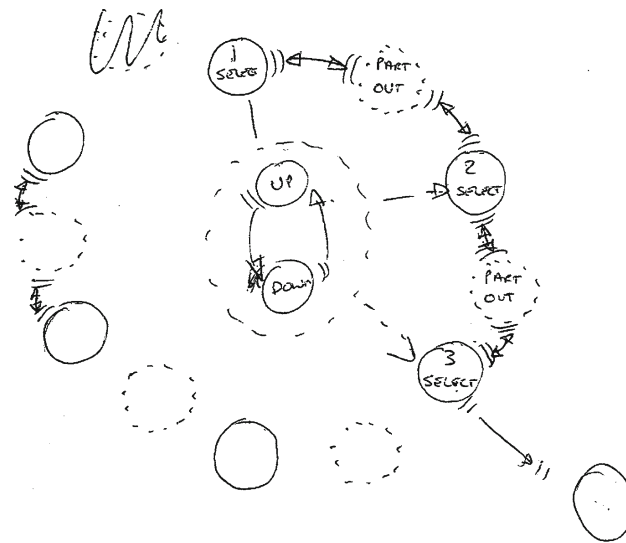


Fig. 15. Early physigrams sketching

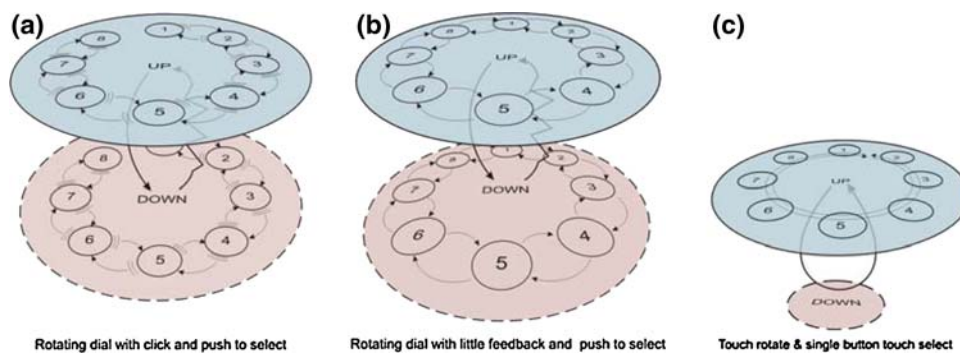


Fig. 16. Physigrams with perspective: a device 1 and 2, b device 3, c device 4

Figure 15 shows an early sketch for device 1. We can see examples of both semantic decisions and human representation ones. Looking round the outside it is clear that the designers were considering whether to represent explicitly intermediate states between the click stops, (labelled ‘part out’), a clear semantic decision. However, also at this stage we see the up/down transition being represented in the middle (rather like a concurrent state diagram in state charts).

This use of space was most complete in a set of alternative 3D perspective physigrams (Fig. 16) that the designers produced. Here they explicitly explained how the layering of the up and down super states conveyed the fact that down was literally pressing down.

It is tempting in a visual notation to try to use the 2D layout to convey explicit semantics, for example, some notations use juxtaposition to mean communication, or left-to-right layout as sequence. Clearly some level of this is useful conveying formal semantics implicitly using the human visual system. However, it is the very fact that the formal semantics of the physigrams left much of the 2D layout *uninterpreted* that makes it available for additional human interpretation. This parallels in notation design, one of the general principles of designing for appropriation: ‘allow interpretation’ [Dix07]. The importance of such ‘secondary notation’ (annotation or variants in notations that do not impact on semantics) has also been recognised in the cognitive dimensions literature providing, *inter alia*, means to add more nuanced human-interpretable aspects to an underlying formal notation [GrP96].

It was perhaps a little surprising that none of the physigrams included photographs or sketches of the visible aspects of the devices. The diagrams did explicitly encode the haptic and tactile differences between devices 2

and 3 and between device 3 and 4, but not the visual difference between device 1 and 2. Device 1 and 2 share an identical physigram, but device 1 is an exposed state device whereas device 2 is a hidden state device. This does not show up in the state diagrams for the physical device, but would do in the mapping between device states and visible states. During discussion it became clear that the physigrams had not been annotated with the visible form (as in Fig. 2a) because the physical devices were there in front of the designers. The visual aspects were immediately obvious and did not need to be formally recorded, however the aspects that were less obvious were more worthwhile to express in the physigram. This was because the physigrams were being used as a live tool during a design meeting. If they were used to communicate between different teams—as when the developers tried to interpret them—it would be more important to give explicit guidance on visual annotations.

8.5. Designers' impressions

It is important to note that generally designers do not normally work this way, it is outside their comfort zone. That said, they were able to work with the concept quite quickly, producing the first finished physigram within about 20 minutes and the new 3D physigram concept within an hour of starting the task.

When initially asked, the design team envisaged using physigrams as a retrospective descriptive tool, probably because this was effectively how they were being used in the exercise. However, after further consideration, the designers suggested that there would be more value in deploying physigrams to describe the interaction early in the design process when the real interaction cannot be prototyped. This would aid communication within the design team and perhaps help individual designers' thought processes, describing and analysing how an interaction should be before the prototyping stage.

The designers also speculated on how it would be to apply this technique to a whole product rather than just a simple dial. They wondered how extra interactions would be conveyed, as separate diagrams or a complete diagram for the product? They also questioned whether interactions would be represented completely separately and whether this actually helps communication between design teams or introduces complications. Effectively they were asking questions about the way a notation could handle both details of the physigrams and also the complexity of the system as whole; questions familiar to those involved in many kinds of formalism.

Considering physigrams as a communication tool for the design process, the designers appeared to feel most comfortable working at level 2, described previously as 'the device with its internal electronics, but unplugged from its application', as evidenced by their representation of device 4. This may have been influenced by the fact that the prototypes used the IE system to link to a Flash animation and were thus very aware of the events generated by the different IE units. This level of analysis helped them make distinctions between the device as prototyped (with all the limitations of the off-the-shelf components) and the device as envisaged in production, so was valuable in that respect. However, as discussed earlier, the disadvantage of this level of analysis is that they were effectively encoding information in the physigram that would not be apparent to a user simply picking up the physical device. The level 1 description would have potentially sensitised the designers to these 'pick up and use' aspects of the device. More work is clearly needed to establish the best form and level of this kind of specification, but the proposed use of layers may be a solution.

9. Discussion

We have used a number of examples to show different ways in which the physical states of a device can interact with the logical states of the system. These have reinforced the importance of distinguishing the two and being able to talk about the, sometimes quite subtle, differences between what might appear to be similar controls.

Each example has dealt with a different property that we have introduced in previous work: exposed state, hidden state, bounce-back, controlled state and compliant interaction. For each of these we have (i) discussed examples informally, then (ii) expressed the examples using parallel state transition networks for the physical and logical states and (iii) given the STNs and their relationship semantics using a formal model. We have introduced the formal model piecewise as each property requires additional elements in the model.

For practical design the variants of STNs would seem more appropriate than the model, although the latter gives the former a more precise semantics. The simpler examples and the relationship between the physical and logical STNs could be dealt with using standard notations and certainly could be translated into state-charts or similar notations. However, as we looked at more complex properties such as bounce-back we had to extend

standard state-transition networks to represent the additional effects. This exposes design issues such as the appropriate use of tension states.

When real designers used the physigrams we found that even the two level distinction between physical ‘unplugged’ device and the logical states was not sufficient. We found we also needed to consider the device at an intermediate level ‘unplugged’ and yet with its internal digital aspects. While we had already had intimations of this in time-dependent events, the additional complexity of the novel prototypes exposed this and additional issues, highlighting the importance of exposing formal work to empirical study.

Possibly most surprising for the designers was the developers’ use of flexibility in the layout of the formal notation to add additional informal interpretations. In retrospect this was fully in accordance with design guidance for appropriation of artifacts. As a general rule this suggests that formal notations should attempt to leave aspects without a formal interpretation, in particular layout; thus leaving these aspects open to human interpretation.

Issues of continuity have arisen repeatedly throughout this paper, both in the standard device examples and in the design case study. Human action and physical device interaction is not simply a matter of ‘events’ occurring and their discrete effects on state. In real life we interact continuously and experience continuous responses; we exert force and feel pressure. However, we also experience discontinuous effects, both with physical devices (when the light switch snaps to a new position) and even more so in digital interactions. This suggests that a deeper semantics based on status–event analysis is still needed in order to map the still discrete formal modelling of this paper into something approaching the physics of real life.

A specific example of continuity in physical interaction was the importance of ‘give’ in several devices. While mentioned obliquely in the literature, this does not appear to have yet had the attention it deserves in allowing exploration of the action potential of devices. Purely digital interactions rarely have this ability, however there are some interaction techniques that have some of this quality. For example in the most recent version of Microsoft Office transparent context-dependent tool palettes become more opaque when the mouse moves towards them.

The haptic feedback of this ‘give’, along with the (only just) perceptible tactile feedback of some devices, and the way physical constraints become a form of dialogue, all point the way to effective use of physicality to guide or constrain digital interactions.

Returning to our two goals. We have obtained new insights and understanding based on the analysis, for example, the issue of ‘give’ above and the way this cast new light on sequential affordances. We have also seen that physigrams have potential within the design process although this requires further study and tool support.

We chose to restrict our scope to the behavioural aspects of the physical device. However, various other physical aspects of the device are important. These include CAD diagrams, which are used extensively in product design; detailed layout [Thi07], the human control loop [E06] and the disposition of the device in its environment [RDR05]. Looking forward it would be valuable to seek ways to link these different aspects. A single notation for all is likely to be cumbersome, but a patchwork of notations and representations could be linked by a unifying abstract semantics. The development of such a unifying semantics would be a major challenge involving representations of continuous action, physical pressure and the physics of the environment itself.

Acknowledgments

This work has been supported by the AHRC/EPSRC funded project DEPTH ‘Designing for Physicality’ (<http://www.physicality.org/>), part of the joint AHRC/EPSRC research initiative ‘Designing for the 21st Century’ (<http://www.design21.dundee.ac.uk/>).

References

- [Ans92] Anson E (1992) The device model of interaction. *SIGGRAPH Comput Graph* 16(3):107–114
- [AvH02] Avrahami D, Hudson S (2002) Forming interactivity: a tool for rapid prototyping of physical interactive products. In: Proc. of the 4th Conf. on Designing interactive Systems: Processes, Practices, Methods, and Techniques (DIS ’02). ACM, New York, pp 141–146. <http://doi.acm.org/10.1145/778712.778735>
- [BSK03] Benford S, Schnadelbach H, Koleva B, Gaver B, Schmidt A, Boucher A, Steed A, Anastasi R, Greenhalgh C, Rodden T, Gellersen H (2003) Sensible, sensible and desirable: a framework for designing physical interfaces, Technical Report Equator-03-003, Equator, 2003. <http://www.equator.ac.uk/>
- [BoV90] Booker S, Vertelney L (1990) Designing the whole-product user interface. In: Laural B (ed) *The art of computer interface design*. Addison-Wesley, Reading, pp 57–63
- [Bux86] Buxton W (1986) There’s more to interaction than meets the eye: some issues in manual input. In: Norman D, Draper S (eds) *User centered system design: new perspectives on human–computer interaction*. Lawrence Erlbaum Associates, Hillsdale, pp 319–337,

- [Bux90] Buxton W (1990) A three-state model of graphical input. In: Proc. of human-computer interaction—INTERACT '90. Elsevier, Amsterdam, pp 449–456
- [CMR90] Card S, Mackinlay J, Robertson G (1990) The design space of input devices. In: Proc. of CHI'90. ACM Press, New York, pp 117–124
- [CMR91] Card S, Mackinlay J, Robertson G (1991) A morphological analysis of the design space of input devices. *ACM Trans Inf Syst* 9(2):99–122
- [Car94] Carr D (1994) Specification of interface interaction objects. In: Proc. of CHI '94. ACM, New York, pp 372–378
- [CoN06] Coutrix C, Nigay L (2006) Mixed reality: a model of mixed interaction. In: Proc. of AVI'06. ACM Press, New York, pp 43–50
- [CoN08] Coutrix C, Nigay L (2008) Balancing physical and digital properties in mixed objects. In: Proc. of AVI'08, the Working Conf. on Advanced Visual interfaces. ACM Press, New York, pp 305–308
- [CuR07] Curzon P, Rukšėnas R, Blandford A (2007) An approach to formal verification of human-computer interaction. *Formal Aspects Comput* 19(4):513–550
- [Deg04] Degani A (2004) *Taming HAL: designing interfaces beyond 2001*. Palgrave Macmillan, New York
- [DiR85] Dix A, Runciman C (1985) Abstract models of interactive systems. *People and computers: designing the interface*. Cambridge University Press, Cambridge, pp 13–22
- [Dix91] Dix A (1991) Formal methods for interactive systems. Academic Press, New York. <http://www.hiraeth.com/books/formal/>
- [Dix91b] Dix A (1991) Status and events: static and dynamic properties of interactive systems. In: Proc. of the Eurographics Seminar: Formal Methods in Computer Graphics. <http://www.hcibook.com/alan/papers/euro91/>
- [DiA96] Dix A, Abowd G (1996) Modelling status and event behaviour of interactive systems. *Softw Eng J* 11(6):334–346 (1996). <http://www.hcibook.com/alan/papers/SEJ96-s+e/>
- [DiA96b] Dix A, Abowd G (1996) Delays and temporal incoherence due to the mediated status-status mappings. *SIGCHI Bull* 28(2):47–49
- [Dix03] Dix A (2003) Getting physical, keynote at: OZCHI 2003, Brisbane, Australia. <http://www.hcibook.com/alan/talks/ozchi2003-keynote/>
- [DFA04] Dix A, Finlay J, Abowd G, Beale R (2004) *Human-computer interaction*, 3rd edn. Prentice Hall, Englewood Cliffs. <http://www.hcibook.com/e3/>
- [Dix07] Dix A (2007) Designing for appropriation. In: Proceedings of BCS HCI 2007, People and Computers XXI, vol 2, BCS eWiC. <http://www.bcs.org/server.php?show=ConWebDoc.13347>
- [DGR07] Dix A, Ghazali M, Ramduny-Ellis D (2007) Modelling devices for natural interaction. In: Proc. of Second Intl. Workshop on Formal Methods for Interactive Systems, FMIS2007, ENTCS. Elsevier, Amsterdam
- [DLF07] Dix A, Leite J, Friday A (2008) XSED—XML-based description of status-event components and systems. In: Proc. of Engineering Interactive Systems 2007 (EIS 2007). Lecture notes in computer science, vol 4940. Springer, Berlin
- [DSG02] Dubois E, Silva P, Gray P (2002) Notational support for the design of augmented reality systems. In: Proc. of the 9th International Workshop on interactive Systems. Design, Specification, and Verification, DSVIS2002. Lecture notes in computer science, vol 2545. Springer, Berlin, pp 74–88
- [DuG07a] Dubois E, Gray P (2008) A design-oriented information-flow refinement of the ASUR interaction model. In: Engineering interactive systems (incorporating EHCI, HCSE, DSV-IS). Lecture notes in computer science, vol 4940. Springer, Berlin
- [DGR07b] Dubois E, Gray P, Ramsay A (2007) A model-based approach to describing and reasoning about the physicality of interaction. In: Proc. of Physicality 2007. UWIC Press, Cardiff, pp 77–82
- [E06] Eslambolchilar P (2006) Making sense of interaction using a model-based approach. Ph D thesis, Hamilton Institute, National University of Ireland, NUIM, Ireland
- [EvG06] Evans M, Gill S (2006) Rapid development of information appliances. In: Proc. of International Design Conf. Design 2006, (Croatia, 15–18 May 2006)
- [Gav91] Gaver W (1991) Technology affordances. In: Proc. of CHI '91. ACM Press, New York, pp 79–84
- [GhD03] Ghazali M, Dix A (2003) Aladdin's lamp: understanding new from old. In: Proc. of 1st UK-UbiNet Workshop, Imperial College London. <http://www.hcibook.com/alan/papers/ubinet-2003/>
- [GhD05] Ghazali M, Dix A (2005) Visceral interaction. In: Proc. of the 10th British HCI Conf., vol 2, pp 68–72. <http://www.hcibook.com/alan/papers/visceral-2005/>
- [GhD06] Ghazali M, Dix A (2006) Natural inverse: physicality, interaction & meaning. In: Let's Get Physical: Tangible Interaction and Rapid Prototyping in, for, and about Design Workshop at 2nd International Conf. on Design Computing & Cognition 2006
- [Gib86] Gibson J (1986) *The ecological approach to visual perception*. Houghton Mifflin, USA
- [GLH05] Gill S, Loudon G, Hewett B, Barham G (2005) How to design and prototype an information appliance in 24 hours—integrating product & interface design processes. In: Proc. of the 6th International Conf. on Computer Aided Industrial Design and Concept Design, University of Delft, The Netherlands
- [GrP96] Green T, Petri M (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *J Vis Languages Comput* 7:131–174
- [GrF01] Greenberg S, Fitchett C (2001) Phidgets: easy development of physical interfaces through physical widgets. In: Proc. of the 14th Annual ACM Symposium on User interface Software and Technology (UIST '01), pp 209–218. <http://doi.acm.org/10.1145/502348.502388>
- [Har87] Harel D (1987) Statecharts: a visual formalism for complex systems. *Sci Comput Program* 8(3):231–274
- [HKC07] Harrison M, Kray C, Campos J (2007/2008) Exploring an option space to engineer a ubiquitous computing system. In: 2nd International Workshop on Formal Methods for Interactive Systems (FMIS 2007), Electronic Notes in Theoretical Computer Science, vol 208. Elsevier, Amsterdam, pp 41–55
- [HKB05] Hartman B, Klemmer S, Bernstein M, Mehta N (2005) d.tools: Visually Prototyping Physical UIs through Statecharts. In: Extended Abstracts of UIST 2005. ACM, New York
- [Har03] Hartson H (2003) Cognitive, physical, sensory, and functional affordances in interaction design. *Behav Inform Technol* 22(5):315–338

- [Ish08] Ishii H (2008) Tangible user interfaces. In: Sears A, Jacko J (eds) The human–computer interaction handbook fundamentals, evolving technologies, and emerging applications, Chapter 24, 2nd edn. Laurence Earlbaum, London, pp 469–487
- [JDM99] Jacob J, Deligiannidis L, Morrison S (1999) A software model and specification language for non-WIMP user interfaces. *ACM Trans Comput Hum Interact* 6(1):1–46
- [Joh96] Johnson C (1996) The evaluation of user interface design notations. In: Proc. of Design, Specification and Verification of Interactive Systems '96. Springer, Berlin, pp 188–206. http://www.dcs.gla.ac.uk/~johnson/papers/chris_jarle/
- [LaM95] Landay J, Myers B (1995) Interactive sketching for the early stages of user interface design. In: Proc. of CHI'95. ACM Press/Addison-Wesley, New York, pp 43–50. <http://doi.acm.org/10.1145/223904.223910>
- [LoH02] Loer K, Harrison M (2002) Towards usable and relevant model checking techniques for the analysis of dependable interactive systems. In: Proc. 17th International Conf. on Automated Software Engineering. IEEE Computer Society, New York, pp 223–226
- [MDS99] Massink M, Duke D, Smith S (1999) Towards hybrid interface specification for virtual environments. In: DSV-IS 1999 Design, Specification and Verification of Interactive Systems. Springer, Berlin, pp 30–51
- [Mil88] Milner N (1988) A review of human performance and preferences with different input devices to computer systems. In: Proc. of HCI88, People and Computers IV. Cambridge University Press, Cambridge, pp 341–362
- [MDB96] Moher T, Dirda V, Bastide R, Palanque P (1996) Monolingual, articulated modelling of users, devices and interfaces. In: 3rd EUROGRAPHICS workshop on design, specification and verification of Interactive systems. Springer, Berlin, pp 312–329
- [Mon86] Monk A (1986) Mode errors: a user-centered analysis and some preventative measures using keying-contingent sound. *Int J Man Mach Stud* 24(4):313–327
- [NiM94] Nielsen J, Mack R (1994) Usability inspection methods. Wiley, New York
- [NCo91] Nigay L, Coutaz J (1991) Building user interfaces: organizing software agents. In: ESPRIT '91 Conf., pp 707–719
- [NCo95] Nigay L, Coutaz J (1995) A generic platform for addressing the multimodal challenge. In: Proc. of CHI'95. ACM, New York, pp 98–105
- [Nor99] Norman D (1999) Affordance, conventions, and design. *Interactions* 6(3):38–43
- [PaP97] Palanque P, Paterno F (1997) (eds) Formal methods in human–computer interaction. Springer, Berlin
- [Pa69] Parnas D (1969) On the use of transition diagrams in the design of a user interface for an interactive computer system. In: Proc. of the 1969 24th National Conf.. ACM, New York, pp 379–385
- [PaG86] Payne S, Green T (1986) Task–action grammars: a model of mental representation of task languages. *Hum Comput Interact* 2(2):93–133
- [PfH85] Pfaff G, Hagen P (1985) (eds) Seeheim workshop on user interface management systems. Springer, Berlin
- [Phi08] Phidgets Inc., 2008. <http://www.phidgets.com/>
- [PSP99] Pierce J, Stearns B, Pausch R (1999) Voodoo dolls: seamless interaction at multiple scales in virtual environments. In: Proc. of the 1999 Symposium on Interactive 3D Graphics, pp 141–145
- [RDR05] Ramduny-Ellis D, Dix A, Rayson P, Onditi V, Sommerville I, Ransom J (2005) Artefacts as designed, Artefacts as used: resources for uncovering activity dynamics. *Cogn Technol Work* 7(2):76–87
- [Rei81] Reisner P (1981) Formal grammar and human factors design of an interactive graphics system. *IEEE Trans Softw Eng SE-7(2)*:229–240
- [Shn83] Shneiderman B (1983) Direct manipulation: a step beyond programming languages. *IEEE Comput* 16(8):57–69
- [Shn84] Shneiderman B (1984) Response time and display rate in human performance with computers. *ACM Comput Surv* 16(3):265–285
- [Smi06] Smith S (2007) Exploring the specification of haptic interaction. In: Interactive systems: design, specification and verification (DSVIS 2006). Lecture notes in computer science, vol 4323. Springer, Berlin, pp 171–184
- [Suf82] Sufrin B (1982) Formal specification of a display editor. *Sci Comput Program* 1:157–202
- [ThH90] Thimbleby H, Harrison M (1990) Formal methods in human–computer interaction. Cambridge University Press, Cambridge
- [Thi07] Thimbleby H (2007) Press On: principles of interaction programming. MIT Press, Cambridge
- [Thi07] Thimbleby H (2007) Using the Fitts law with state transition systems to find optimal task timings. In: Pre-Proc. of Second Intl. Workshop on Formal Methods for Interactive Systems, FMIS2007. <http://www.dcs.qmul.ac.uk/research/imc/hum/fmis2007/preproceedings/FMIS2007preproceedings.pdf>
- [UIM92] UIMS (1992) A metamodel for the runtime architecture of an interactive system: the UIMS tool developers workshop. *SIGCHI Bull* 24(1):32–37
- [UIJ05] Ullmer B, Ishii H, Jacob R (2005) Token+constraint systems for tangible interaction (with digital information). *ACM Trans Comput Hum Interact* 12(1):81–118
- [WDO04] Wensveen S, Djajadiningrat J, Overbeeke C (2004) Interaction frogger: a design framework to couple action and function. In: Proc. of the DIS'04. ACM, New York, pp 177–184
- [WiH00] Willans J, Harrison M (2001) Verifying the behaviour of virtual world objects. In: Palanque P, Paternó F (eds) Proc. of DSV-IS'2000. Springer, Berlin, pp 65–77
- [Wüt99] Wüthrich C (1999) An analysis and model of 3D interaction methods and devices for virtual reality. In: Proc. of DSV-IS'99. Springer, Berlin, pp 18–29
- [YGS89] Young R, Green T, Simon T (1989) Programmable user models for predictive evaluation of interface design. In: Proc. of CHI'89: Human Factors in Computing Systems. ACM Press, New York

Received 13 March 2008

Accepted in revised form 30 October 2008 by A. Cerone, P. Curzon and D.A. Duce

Published online 6 December 2008