Chapter 7
Activity Modelling for Low-Intention Interaction

Alan Dix

Abstract When modelling user interactions, we normally assume that the user is acting with intention: some very explicit such as opening a valve in a nuclear power station, others more tacit, hardly needing any thought, for example tipping a tablet to turn a page. However, there are also a range of system behaviours that make use of unintentional user actions, where the user acts, but the system decides that the action has meaning, and how to make use of that meaning. Again, these may operate on a variety of levels from 'incidental interactions', which operate entirely without the user realising, perhaps subtle changes in search results based on past activity, to more 'expected interactions' such as automatic doors that open as you approach. For intentional interaction, there is long-standing advice—making sure that the user can work out what controls do, where information is, interpret the available information, receive feedback on actions—and also long-standing modelling techniques. Low-intention interactions, where the system has more autonomy, require different design strategies and modelling techniques. This chapter presents early steps in this direction. Crucial to this is the notion of two tasks: the sensed task, which the system monitors to gain information and the supported task, which the system augments or aids. First, this chapter demonstrates and develops techniques in the retrospective modelling of a familiar low-intention interaction system, car courtesy lights. These techniques are then applied proactively in the design of a community public display, which is now deployed and in everyday use.

A. Dix (✉)
School of Computer Science, University of Birmingham, Birmingham, UK
e-mail: alan@hcibook.com

A. Dix
Talis Ltd. Birmingham, Birmingham, UK

B. Weyers et al. (eds.), *The Handbook of Formal Methods in Human-Computer Interaction*, Human-Computer Interaction Series,
DOI 10.1007/978-3-319-51838-1_7

7.1 Introduction

Mostly, when modelling user interactions, we assume that the user is acting with intention. Indeed, the presence of a goal is central to Norman's (1990) influential 'seven stages' model. Traditional hierarchical task analysis also starts with a top-level goal, which then leads to a set of tasks to achieve that goal and sub-tasks of those top-level tasks. In a more dynamic fashion, means-end analysis decomposes by creating sub-goals when plans encounter impasses. Sometimes, these intentional actions are very explicit, for example opening a valve in a nuclear power station to alter pressure in the containment vessel; some are 'implicit', hardly needing any thought, for example tipping a tablet to turn a page.

However, there is also a range of system behaviours that make use of unintentional user actions, or to be precise where the use made by the system is not the primary intention of the action. The most extreme are '*incidental interactions*'. In these situations, the user's intention and actions are focused on a primary goal. Because of these actions, the system is able to gather some direct or sensed information, which can then be used to help the user or other agents achieve some secondary goal. Somewhere between these and fully intentional interaction are '*expected interactions*', for example walking into a room and the automatic light coming on. You usually enter the room because you want to be inside, but expect that the light will come on and would be surprised if it did not.

For intentional interaction, there is long-standing advice: make sure that the user can work out what controls do, where information is, interpret the available information and receive feedback on actions (e.g. Nielsen's (1993) ten heuristics or Shneiderman's (1998) golden rules). While some of these design principles and heuristics need interpretation by a graphical, interaction or user experience designer, others can be operationalised in well-known formal models and formalisations of usability principles (e.g. variants of visibility in Dix 1991).

Low-intention interaction is more problematic, as it is not so much a matter of presenting the user with controls and behaviours that are as clear as possible, but instead interpreting the user's actions performed for a variety of other purposes.

This chapter shows how a level of activity modelling can be used in order to ascertain actions or signs that may already be sensed, or where additional sensors can be added, so that this provides sufficient information for informed system action. Context-aware systems do this by creating very explicit system models of user actions. This chapter, in contrast, will focus on designer models of user activity, which can then be analysed to reveal potential triggers for automated action. Crucially, these models need to adopt some level of probabilistic reasoning, although this may be qualitative.

This chapter starts by explaining in more detail what is meant by 'low-intention interaction', using a series of examples, and this includes the two tasks (sensed and supported) mentioned above. This is followed by a short review of related literature including notions of implicit interaction, natural interaction and architectures and tools to support the design and construction of sensor-rich systems. This leads into a

further discussion of design issues for low-intention interaction, notably the way user models do not necessarily embody the same notions of transparency and feedback of intentional systems, and the implications for privacy. Finally, this is brought to bear on the explicit design of low-intention systems centred around the separate analysis of the sensed task (in order to determine what can be known of the users' actions) and of the supported task (in order to determine what are desirable interventions). The initial motivating example will be car courtesy lights (previously described in Dix et al. 2004; Dix 2006), as the technique is most well suited for non-safety critical applications. However, it is then applied to the design of a deployed public display system, which has since been in operation for several years.

The underlying methods used in this chapter date back over ten years and parts draw heavily on early work in incidental interaction (Dix 2002), and its incorporation in the author's HCI textbook (Dix et al. 2004) and online material (Dix 2006) on design methods for incidental interaction. Some of the concepts and methods are developed further and made explicit, but the principal contribution of this chapter is the reporting of how the techniques were applied in practice in the design of TireeOpen, the Internet-enabled open sign.

7.2 What Is Low-Intention Interaction?

7.2.1 Intentional and Low-Intention Interaction

As noted, traditional user interaction principles have focused on the user being in control of precisely what the computer system does. This is at the heart of direct manipulation (Shneiderman 1982; Hutchins et al. 1986); the system state is represented faithfully in the user interface, and as the user modifies the representation, the system immediately updates accordingly. Norman's seven-stage model of interaction starts with a goal, and the user translates this into actions on the system and then evaluates the results of the action. Users are assumed to know what they want to do, and the computer's job is to perform the actions requested as reliably and transparently as possible.

Intentional systems may use intelligent algorithms. For example, the Kinect uses complex vision processing to work out your body position and gestures, but this is in order to be able to accurately understand your intended movements, say to swing a golf club. The goal and the intent lie clearly with the user, and the system merely works to enact or interpret the user's intention.

In contrast, some autonomic systems operate at such a level that the user may be unaware that they are doing anything at all. For example, an intelligent heating system may use mobile phone GPS and other sensors to work out when you are due home, whether you have been walking through the rain, doing a workout at the gym or merely driving back. If the heating system is good enough at predicting the right temperature, you may be completely unaware that it has been making adjustments.

In such systems, the user has no little or no explicit intention; if there is intent, it is on the system's part.

7.2.2 The Intentional Spectrum

Incidental interaction was coined to describe an extreme form of low-intention interaction:

> Incidental interaction—where actions performed for some other purpose or unconscious signs are interpreted in order to influence/improve/facilitate the actors' future interaction or day-to-day life (Dix 2002).

In fact, there is a spectrum with explicitly intended actions at one end and incidental interaction at the other extreme (see Fig. 7.1). In between are 'expected' system actions, for example if you work in an office building with automatic lights, you expect the lights to turn on as you enter a room, even though you do not explicitly flick a switch.

Near to the intentional end are actions such as tipping an e-book reader to turn a page as well as very explicit actions such as pressing a button. At this intentional end, we find actions that are invoked entirely by the user, but are low awareness as they are automatic or even autonomic; when reading a (physical) book you are rarely explicitly aware of turning the page, similarly when using a tool you are often focused on the work at hand, not the tool itself, Heidegger's (1927) 'Ready at hand'.

7.2.3 Examples of Low-Intention Interaction

Many research systems include aspects of low-intention or incidental interaction.

The Pepys system in Xerox EuroPARC used infrared badges to track researchers in the office and then created automatic diaries at the end of each day (Newman et al. 1991). Here, the primary, intentional task was simply to walk to a colleague's office, but, incidentally, the location was tracked and the diary produced.

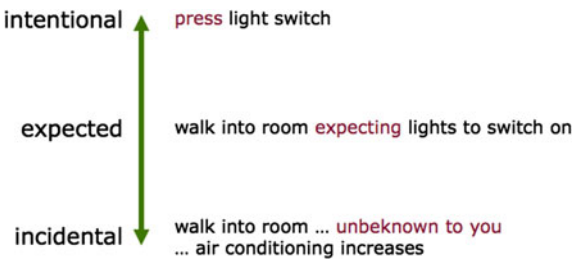

Fig. 7.1 Continuum of intentionality (from Dix et al. 2004)

One of the defining ubiquitous computing applications was MediaCup (Beigl et al. 2001; Gellersen et al. 1999). MediaCup added a small sensor pack to the base of ordinary mugs measuring pressure (whether the cup was full or empty), temperature and tip sensors. As the cup owner filled and drank their coffee, the system collected the sensor data and relayed this to others who could then build an idea of the cup owner's activity and availability. Again, the primary (sensed) task was drinking the coffee, but, incidentally, other people were able to improve their interpersonal interactions.

In the author's own work in the late 1990s, onCue provided an intelligent desktop task bar (Dix et al. 2000b). Whenever the user cut or copied things into the clipboard (primary task), onCue analysed the clipboard contents and then suggested possible additional things that could be done using the clipboard content on the Internet or desktop. For example, when you copied a postcode, onCue would suggest Web-based mapping services, or when you copied tabular data, onCue would suggest Web graphing tools or copying into Excel on the desktop.

Even in the age of the Internet of Things (IoT), Internet-enabled mugs are not common. However, incidental interactions are common in day-to-day life.

As you walk towards a building, the doors open, lights go on as you enter a room, toilets flush as you leave the cubicle, and when you get into your car, the courtesy lights go on. In each case, your primary task is simply going in or out of a room, car or toilet cubicle, but the system senses aspects of your primary activity and then performs additional actions aimed to help you. Location-aware apps in phones and wearables use our movements in the environment to enhance secondary goals: telling friends where we are, recording fitness information.

To some extent, this trend is still accelerating as the availability of low-power, small-sized and, crucially, cheap sensors and networking is only just making the visions of 1990s ubiquitous computing commercially possible (Greenfield 2006).

However, in the purely digital domain, where sensing is simply a matter of logging digital interactions, these interactions abound. When we use a shopping site, our primary task may be to buy a particular book or device, but incidentally, the data collected is used to enhance recommendations for other shoppers. When we search, our preferences for clicking through on certain topics (primary task) are often analysed to improve subsequent search result ranking. Possibly less welcome, we also know that our visits to many sites are tracked, sometimes by hundreds of tiny 'beacons', which are then used to gather marketing information and channel particular advertisements to us.

7.2.4 Intentional Shifts

We have seen that there is a continuum between incidental interaction, where users may have no awareness that information is being sensed or the way this is enhancing their interactions, to fully intentional interaction, where users explicitly control a system to fulfil their goals.

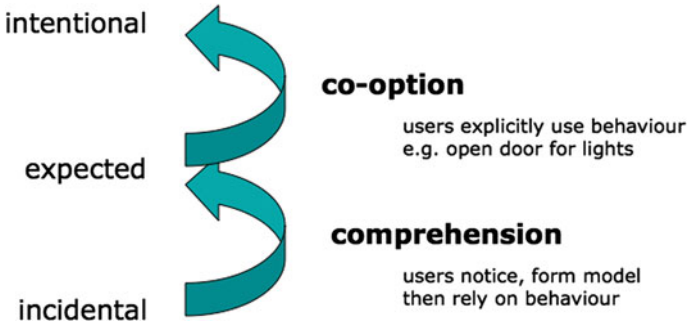

Fig. 7.2 Fluidity of intentionality (from Dix et al. 2004)

Sometimes, parts of a system, or even our own actions of which we are normally unaware, may become more apparent. This might be because some part of the system does not behave seamlessly. Heidegger's (1927) 'breakdowns' occur precisely when more tacit actions are forced into conscious attention, for example if a hammer head works loose. At this point, one shifts into a fully intentional form of interaction.

Even where a system is working correctly, we may become aware of its behaviour. As we begin to understand the rules of sensor-based systems, actions that were entirely below awareness (incidental) may become expected. For example, if the lights in a building operate automatically based on sensors, you may not explicitly intend the lights to switch on when you enter, but you certainly do not expect to be left in darkness.

Once we understand the rules well enough to be 'expected', we may co-opt the system behaviour to exert explicit control over what were originally intended to be incidental interactions. For example, you might open and close a car door to get the courtesy lights to turn on for a longer period, or wave your arms to activate the movement sensor if you want the lights to turn on (see Fig. 7.2).

7.2.5 Two Tasks

As mentioned previously, it is important to recognise that there are two different user tasks to consider:

sensed task—the task related to the user's primary goal during the behaviour that is being sensed by the systems and

supported task—the task that is in some way supported or enhanced by the information inferred from the sensor data.

Sometimes, these are entirely different tasks, for example with the MediaCup (Beigl et al. 2001; Gellersen et al. 1999), the sensed task is about drinking coffee

while the supported task is about meeting at appropriate times. However, in other cases, it may be that these are part of the same overall activity (indeed, in the car courtesy light, in the next section, this is precisely the case). Even when the two are actually the same or related tasks, we are using the task analysis in different ways.

In the case of the sensed task, the user's primary goal is not necessarily being supported, but enables us to interpret the sensors, turning raw data into behavioural information. For example, during normal use, if a car door is opened after the car has been stationary for a period, it is likely that someone has got into the vehicle.

Sometimes, this sensing is purely passive, but we may choose to modify the systems around this sensed task. Where the sensed data is insufficient to interpret behaviour, we may choose to add sensors, for example adding an infrared movement sensor to a car interior to be sure that passengers are present. Alternatively, we may choose to modify the actual user interaction on the sensed task.

For example, consider a library website. We have two design alternatives:

(i) show full information as lots of book 'cards' in a large scrollable page, rather like Pinterest (https://about.pinterest.com/).
(ii) show shorter descriptions (title + teaser) but where the full book details appear as a pop-up when the user hovers over or clicks an item.

Let us assume that you decide (i) is more usable, but that the difference is slight. However, from (ii), it is far easier to identify the user's interests. You may then deliberately decide to adopt (ii), even though it is slightly less usable for the sensed task, because you know that by having more information about the user, the system is better able to make suggestions. In other words, you may choose to trade usability between the sensed and supported tasks.

Turning to the supported tasks, there are various ways in which this support can occur.

While the sensing may be low intention, the information gathered by this may be explicitly presented to the user. For example, onCue monitors the user cutting and copying to the clipboard (the sensed task) and infers the kind of thing in the clipboard (e.g. postcode, personal name, table of numbers); this is all automatic, without user attention (Dix et al. 2000b). However, it then alters the toolbar to show actions that the user can perform using the clipboard contents. The user explicitly interacts with the toolbar; that is the supported task is intentional.

In other cases, system modifications to the supported tasks may also be low attention and/or low intention. For example, the order of search results in the library system may be subtly altered based on inferred interest, or the heating in the room may adjust to be slightly warmer when you are sitting and cooler when you are busily moving around.

Note that low intention and low attention are related but different properties. The level of intention is about the extent to which the user controls the initiation of actions, whereas levels of attention are about how much conscious focus is directed towards actions and their results. A system action might be autonomous and

unexpected (not the user's intention), but still very obvious and salient (high attention). Likewise, a user's action might be fully intentional but low awareness, for example drinking from a cup of tea while chatting.

7.3 Frameworks and Paradigms

There have been a number of models and frameworks that, in different ways, help understand, analyse, design or construct sensor-rich user interactions. In this brief overview of the most related literature, we will first look at various design concepts in the area, followed by a more in-depth analysis of the notion of 'naturalness' as this is closely related to, but distinct from, low attention. Finally, we look at some of the related architectural frameworks and modelling notations.

7.3.1 Design Concepts

In the 1990s, as ubiquitous computing began to mature, the concept of *context-aware computing* emerged (Schilit et al. 1994; Schmidt 2013). Whereas, in traditional computer applications, user input was interpreted solely in terms of the state of the system, context-aware systems looked to the environment or context. Dey defined this context as:

> any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves (Dey 2001).

Schilit et al.'s (1994) early identification of context-aware applications was inspired by PARCTAB, a deployment of Olivetti infrared tracking badges deployed at EuroPARC (Want et al. 1995). More generally, while other kinds of context were considered, mobility and location were often the defining contexts for early work in the area (Dix et al. 2000a). This in turn led to a number of related interaction-focused design concepts.

Schmidt developed the design concept of *implicit interaction* partly to deal with interactions such as tipping a mobile device to turn a page or move a map.

> Implicit human computer interaction is an action, performed by the user that is not primarily aimed to interact with a computerized system but which such a system understands as input (Schmidt 2000).

In many ways, this can be seen as a precursor to the recent focus on, so-called, *natural user interfaces* or NUIs (Wigdor and Wixon 2011). NUIs are characterised by the detection of ordinary human actions such as gaze direction, body movement, or touch, for example touch tables or Kinect games. The use of the term 'so-called'

at the start of this paragraph is because the 'naturalness' of NUIs is often challenged; while the actions are natural ones, the ways in which these are interpreted are often far from natural (Norman 2010).

Ju and Leifer (2008) developed a design framework around Schmidt's notion of *implicit interaction*. The issue of attention is important in their framework, which is particularly focused around two interaction distinctions: attentional demand (foreground vs background) and initiative (reactive vs. proactive).

The concept of *Kinetic User Interfaces* (KUI) emerged during the uMove project (Pallotta et al. 2008) inspired partly by the author's work on *incidental interaction* (Dix 2002). KUIs are where either direct bodily movement, or the user's movement of objects in the environment, is sensed and used for 'unobtrusive' interactions. Like implicit interaction, KUIs embody a particular kind of low-intention interaction.

Wilde et al. (2010) reviewed interaction patterns for pervasive systems based on Alexander et al.'s (1977) architectural pattern language and its uses within software engineering (Gamma et al. 1995). Wilde et al. break down pervasive interaction patterns into three broad classes: interactions with mobile systems, intelligent environments and collaborative work, although many pervasive interactions will of course include elements of each. Forbrig et al. (2013) also consider models and patterns for smart environments building on their concept of *supportive user interfaces*. The patterns are more domain specific than Wilde et al.'s, focusing on smart meeting rooms as an example application area.

Much of the early conceptual work in ubicomp and context-aware computing was primarily descriptive or focused on structuring implementation. The *expected, sensed and desired* framework (ESD) sought to transform much of the practical experience of creating instances of context-aware systems into a more structured ideation and early process (Benford et al. 2005). Expected movements are those that a person might naturally do (e.g. twist a screen, or move in the environment to see something better); sensed movements are those that can be detected by some sort of existing, or easy to add, sensor; and desired movements are the actions you might like to perform (e.g. zoom a display to see more detail).

By making this distinction, it is possible to compare them, for example identifying expected movements that are not sensed suggesting the potential for adding new sensors, or expected movements that can be sensed, which might be used to control desired actions.

Note, in incidental interaction terms, the expected and sensed movements are primarily concerned with the primary goal (sensed task), whereas the desired movements concern the supported task. Also, while the earliest work leading to this chapter predated ESD, aspects of the formulation later in this chapter are influenced very much by ESD.

7.3.2 *Low Intention and Naturalness*

The concepts of implicit interaction and natural user interfaces (NUIs) are particularly relevant to low-intention interaction. Figure 7.3 shows some of the distinctions they raise.

At the top left are the most artificial actions, for example when you come to a new interface and have to work out which of the new icons or menu choices you need to use. However, with practice, they become second nature (bottom left), and you may not even be aware that you are doing them.

On the far bottom right are the 'most natural' interactions, those that you never have to think about at all. These may be truly instinctive; one particular case of this is where there is a 'natural inverse' (Ghazali and Dix 2006) (e.g. push/pull, twist left/twist right); in such cases, users automatically do the opposite action when they 'overshoot' a target, for example correcting on a steering wheel. Often they are themselves learnt, for example swinging your arm is in a sense an unnatural action to control a computer simulation, but when faced with a Kinect and an image of a golf ball, it recruits already learnt physical actions.

Many NUI interactions are not so obvious. Early research on touch tables found that given free choice, most people used the same dragging, twisting and stretching actions, but beyond these, there was little agreement. However, after a period of use, even the less intuitive gestures become automatic, and perhaps more quickly learnt than abstract icons, as they are able to employ kinesthetic or other forms of sensory memory.

NUIs cover the whole of the right-hand side of the figure, whether immediately obvious and later learnt, or so immediate you never know you are doing them. The early examples of implicit interaction are in this space, notably the turning of pages by tipping a device.

From a low-/high-attention point of view, actions may be very artificial (e.g. shifting gears in a car), but so automatic that you are unaware you are doing them,

However, when we consider the sensed and supported tasks of incidental interaction, in fact, even very explicit and artificial actions may be part of a wider low-intention interaction.

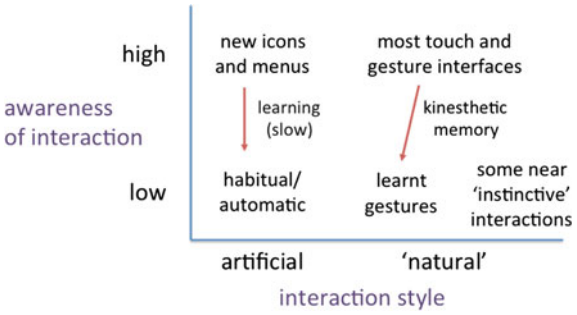

Fig. 7.3 Various forms of natural interaction

Consider an example of a sensed task. When you browse on Amazon, this is an explicit action and may include some high-awareness actions (what shall I search for) and some low-attention learnt interactions (clicking through for more details). However, all of this may be used by Amazon to build its profile of you; the actions are explicit, but the way information is collected about them is not.

Similarly, in the supported task, there may be entirely autonomous actions that the system executes based on your past behaviour (top right of Fig. 7.4, Ju and Leifer's (2008) 'proactive' initiative). For example, an intelligent heating system might automatically set the temperature just right for when you enter your home in the evening. However, you might also execute an explicit action, the effect of which is modified based on previous behaviour or current context (bottom right Fig. 7.4, Ju and Leifer's (2008) 'reactive' initiative). For example, if you hit 'cooler' on the heating system, this might be interpreted as a short blast of cool air, but gradually returning to normal if the system knows (say from phone sensors) that you have just come in after running hard, but behave differently if it is pressed after you have been at home for a while.

That is the key difference for the supported task is whether the action is fixed or contextual based on previous sensed behaviour.

7.3.3 *Architecture and Modelling*

Dey's definition of 'context' quoted above was influenced by the development of the context toolkit (Salber et al. 1999), which was important in signalling the movement from more specific bespoke developments towards more principled design and reusable architecture. The author's own work in the area included the development of the onCue context-sensitive toolbar, which was built on top of an agent-based framework, *aQtive space*, specifically architected to support context-sensitive interaction (Dix et al. 2000b).

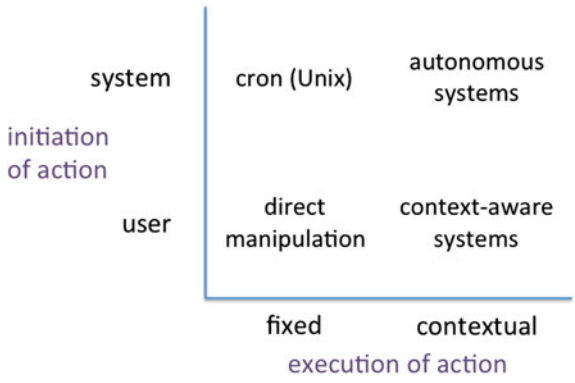

Fig. 7.4 Supported actions

Fig. 7.5 Context description from Schmidt (2000)

```
<context_interaction>
    <context>
        <group match='one'>
            sensor_module.touch
            pilot.on
        </group>
        <group match='none'>
            sensor_module.alone
            pilot.pen_down
        </group>
    </context>
    <action trigger='enter' time='3'>
        pilot.notepad.confidential
    </action>
</context_interaction>
```

The aQtive space framework itself was built directly upon earlier formal modelling concepts of *status–event analysis* (Dix and Abowd 1996). While many intentional user interactions are *event* based (e.g. pressing a key) as are low-level computer implementations, sensor data are more often a *status*, that is there is always an underlying value even if it is only sampled periodically. Work following on from this has included an XML component-based notation (Dix et al. 2007).

Schmidt's work on implicit interaction also included an XML-based specification notation (Fig. 7.5), and other work in the area has included discrete event modelling (Hinze et al. 2006) and context-aware architecture derived from MVC (Rehman et al. 2007).

There have been a number of projects which have created combinations of models, tools and evaluation methods for pervasive systems or smart environments.

In addition to the Kinetic User Interface approach, the uMove project led to the creation of a framework including conceptual modelling, architectural design and implementation tools as well as an evaluation method, IWaT (Interactive Walk-Through) (Bruegger et al. 2010; Bruegger 2011).

Wurdel addressed similar broad goals, but focused more on task modelling using a notation CTML (Wurdel 2011). CTML is an extension to CTT (Paterno 2012) with additional primitives to deal with both actions that are not explicitly addressed to the system (but may be sensed by it), and autonomous system behaviours including those that directly or indirectly affect the user. The resulting task models are used partly to drive model-based system development and partly as input to hidden Markov models.

Another very similar project by Tang et al. (2014) creates a variety of OWL-based notations and integrated design and development tools for pervasive applications including task specification and service design.

7.4 Modelling Low-Intention Interactions

In order to explore ways to formally model these behaviours, we will use a rational reconstruction of the design of car courtesy lights (originally developed in Dix et al. 2004; Dix 2006). These usually include an explicit off/on switch, but also turn on automatically at times.

7.4.1 Modelling Process

We will not create a new notation, but instead augment standard task and state description formalisms. The task descriptions here will be simple scenarios augmented by state-space models, but richer descriptions could be used such as HTA (Shepherd 1989), CTT (Paterno 2012) or CTML (Wurdel 2011).

The critical steps are to:

(i) annotate the supported task to see where augmentation or assistance would be useful. In the examples, this is in the form of + or – to indicate how desirable or undesirable a particular assistance would be.
(ii) annotate the sensed task to see where existing or new sensors could give useful information and to assess the likelihood that sensed state matches the target context in the real world.

Often sensors are attached to objects in the environment; so for (ii), it can be useful to model explicitly the states of physical (or virtual) objects.

Note that the assessment of desirability in (i) is a core design decision. It may be a matter of discussion and it may include more or less desired states as well as hard and fast ones. This is particularly important as sensed input in (ii) is rarely precise. This imprecision may be because of limitations in sensors, because sensed user actions may be typical but not guaranteed, or because the target world state may not be directly able to be monitored, meaning some proxy has to be sensed instead. By having an explicit notion of both desirability and precision, we can ensure that sensors are chosen to ensure the best behaviour in critical situations, with leeway in the less critical ones (Fig. 7.6). For example, we may be able to adjust threshold values to ensure correct behaviour in the critical areas (ticks and crosses) but simply achieve ‘good enough’ results in the less critical areas (grey ‘leeway’).

As we work through the examples adding annotations to the task models, we will develop a form of secondary notation (Green and Petre 1996). In particular, when looking at the supported task, we will use + and – to denote the desirability or otherwise of a particular intervention. However, these are not intended to be a closed set of annotations; indeed, the ‘bomb’ annotation in Fig. 7.7 was added when the example was being used in class and the need arose to highlight particularly problematic situations.

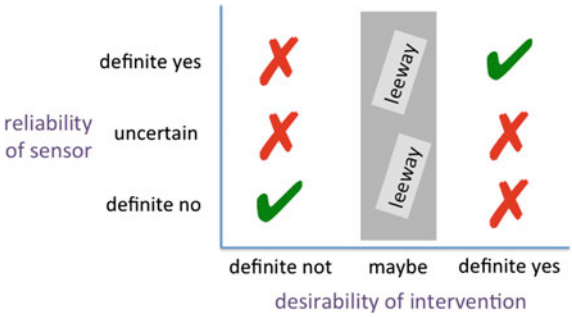

Fig. 7.6 Matching desirability of intervention in the supported task (support, augmentation, autonomous action) with reliability of sensing in the sensed task

1.	deactivate alarm	0	
2.	walk up to car	++/– –	is this safe?
3.	key in door	–	
4.	open door & take key	+	
5.	get in	++	
6.	close door	0	
7.	adjust seat	+	
8.	find road map	++	
9.	look up route	+++	
10.	find right key	+	
11.	key in ignition	–	
12.	start car	0	
13.	seat belt light flashes	0	
14.	fasten seat belt	+	
15.	drive off	– –	safe? legal?

Fig. 7.7 Car courtesy light—getting into the car (from Dix et al. 2004) + signifies would like light on,
– signifies would like light off, 'bomb' signifies that safety is a major issue

7.4.2 Car Courtesy Lights

In the case of the car courtesy light, the sensed task and the supported task are identical. Figure 7.7 shows the task of getting into the car (there would be other tasks such as stopping and getting out, sitting in the car to have a picnic). The main steps are listed and against each one whether or not the courtesy light is wanted.

The pluses mean it would seem good to have it on, the more pluses, the more beneficial. The minus signs show where it would be best to have it off, and the bomb situations where there could be safety issues. Step 15 is obviously problematic and it may be distracting if the courtesy light stays on while driving, and in some places may even be illegal. Step 2 is also marked as potentially unsafe as it might allow a mugger to see which car you are going to, although could also be useful in helping you find your car. Cars vary on whether they choose to turn lights on in these circumstances.

In addition, we ought to look at the scenarios such as when the driver is leaving the car, picking up or dropping off passengers. However, on the whole, the lights want to be (a) on when you are sitting in the car, (b) not left on for long periods

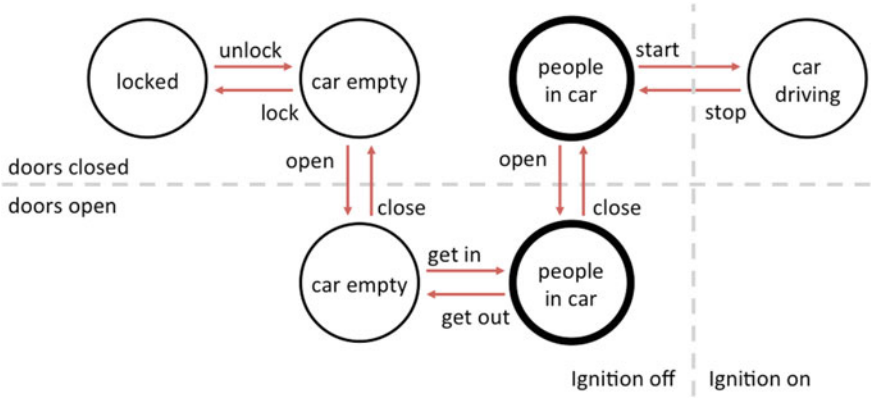

Fig. 7.8 States of the physical car

when you are not in the car (so as not to drain the battery), but (c) not on when you are actually driving (for safety). Of these, (c) is an absolute requirement, (a) is a 'good to have' (you can always turn them on or off explicitly) and (b) is a strong requirement, although the precise time can be quite relaxed.

One could add sensors to the car, for example a PIR (passive infrared) sensor in the car to detect movement so that you can tell if it is occupied. However, cars already have a range of sensors that are reused by the manufacturers to produce automatic lighting. Notably, there is typically a door sensor that triggers a warning if the car is driven without closing the doors properly, and also it is possible to detect the power when the ignition is turned on.

Figure 7.8 shows the main states of the car in terms of people's presence (omitting unusual cases such as people locking themselves inside). These are divided into the states that can be distinguished by the two available sensors.

Note that it is the two states in bold, when people are in the car, which are where we would like the light to be on.

It is possible to tell with certainty when the car is being driven as the ignition is on, so this is a reliable sensor to use to ensure that the lights are not automatically turned on. Most cars gradually dim the lights as soon as the ignition is turned on.

However, it is impossible to tell from these sensors the difference between opening the car door and then realising you have forgotten something and going back into the house to get it, unless you lock the car. Most cars simply use a timer for this, turning the light on when the car doors are first opened, and then off after a period. This is based on the assumption that the sojourn in the 'people in car' state before transitioning to 'car driving' is brief. The timer means that the light is likely to be on most of the time during this state (fulfilling the soft requirement), while also ensuring the light is not on for too long in the 'car empty' state (hard but variable time requirement).

It is clear that having access to the car lock state or a PIR would be useful, as the latter is often fitted as part of the car alarm system. Presumably, the alarm

subsystem in most cars is relatively isolated from the rest of the car subsystems for security reasons and may often be fitted by third parties, so cannot be accessed by the courtesy light subsystem. However, it would not be hard to imagine a future in-car IoT framework to allow more creative use of sensors.

7.5 Into Practice: The Internet-Enabled Shop Open Sign

The interaction styles and techniques in this paper were put into practical use in the design of a community public display system on the Isle of Tiree. This was produced largely as part of the biannual Tiree Tech Wave series of technology/maker meetings (tireetechwave.org). The system is a long-term 24/7 deployment, and the overall data and public display architecture are described in detail in Chap. 4.

In this section, we will focus on the design of two particular elements of this, the Internet-enabled open sign and LED ticker-tape display, which provide sensor input to the data infrastructure. We start with an initial 'provotype'-style (Boer and Donovan 2012) design concept (the fish and chip van that tweets) and then move on to the Internet-enabled open sign, which has been in operation now for several years.

7.5.1 Concept—The Chip Van That Tweets

At the first Tiree Tech Wave, several themes came together around the creation of a concept mini-project, the 'Chip Van That Tweets' (Dostal and Dix 2011).

There was a fish and chip van positioned quite close to the location of the event. Many island businesses are run by a single individual and so can be fragile: if the individual or one of their family is ill, or there is some other kind of emergency, the shop or van may be late opening or close early.

On the mainland, if you went to the fish and chip shop, but found it unexpectedly shut, there would be another close by. On the island, you may have driven perhaps up to ten miles (15 km), over bumpy roads, and there is nowhere else to go. Occasionally, people would ring friends who were within sight of the chip van to ask whether it was open before setting off.

As a light-hearted exemplar, the participants created a prototype to address the issue.

A near full-size mock-up of the van front was constructed out of cardboard. On the flap at the front of the van, an Arduino was positioned with a tilt switch. When the van flap was opened, the tilt switch was activated and the Arduino connected to a mobile phone and sent a tweet via SMS '#tireechipvanopen'. When the van flap closed, the Arduino sent the tweet '#tireechipvanclosed'.

Software that could run on an islander's computer at home listened using the Twitter API and when it saw '#tireechipvanopen' or '#tireechipvanclosed', it sent a

Fig. 7.9 Tiree chip van—sensed task (owner)

0. Serving chips
1. drive to van
2. enter into van (opens side door)
3. prepare for evening:
   turns on power, lights, deep fat fryer
4. open serving flap (at opening time)
   ** sensed by system – tweets #tireechipvanopen
5. serving customers
   take order, fry food, wrap cooked food, take money
6. close serving flap (at closing time)
   ** sensed by system – tweets #tireechipvanclosed
7. tidy up
8. leave van (close side door)
9. go home

Fig. 7.10 Tiree chip van—supported task (customer)

0. Buy and eat chips
1. decide would like fish and chips
2. check if open
   2.1 check current time and opening hours (old)
   2.2 look at chip van model on mantlepiece (new)
3. drive to chip van
4. buy fish and chips (if open)
   or
5. disappointed and hungry (if closed)
6. drive home

message to another Arduino, which was connected to a model fish and chip van, maybe sitting on the potential customer's mantlepiece. A small motor then opened or closed the model to match the real-world fish and chip van.

This is a form of incidental interaction, and Figs. 7.9 and 7.10 show the sensed and supported tasks. While the chip van owner is opening and serving the focus is always on preparing and opening the van, incidentally this leads to improving the customer's interaction efficiency (at 2.2) and experience (4 rather than 5!).

7.5.2 TireeOpen—The Internet-Enabled Open Sign

The chip van that tweets was created as a concept project. It exemplified many important and very practical issues of island life and also was interesting technologically, using Twitter as middleware and an 'Internet of things' architecture. However, it was playful and probably slightly over the top in terms of technology.

Although the technology for the prototype was largely working, there was a gap as the correct kind of phone was not available, so the step between the Arduino detecting the open serving flap and the tweet SMS being sent was emulated by the Arduino flashing an LED and the tweet being sent by hand.

In one way, it would have been a small step to deploy this as a form of technology probe (Hutchinson et al. 2003), but the patchy mobile signal on the island would probably render even the SMS tweeting unreliable.

This last point is not inconsiderable for the practical application of Internet of Things—it only works if you have the Internet, or at least some data communications.

The Cobbled Cow café is based at the Rural Centre on Tiree, the venue of the Tiree Tech Wave. The Cobbled Cow has some of the same issues as the chip van (small family business, illness or other unexpected events can lead to late opening, etc.); however, unlike the chip van, Wi-fi is available.

The closest equivalent to the chip-van serving flap is unlocking the café door, but it is harder to add a sensor to a door lock, than to a flap opening, and it would mean modifying the physical fabric of the café. However, there is also an LED 'open' sign in the window, which is switched on as part of the opening up process (see Fig. 7.11). This already has low-voltage power, hence easy and safe to modify. Modifying the open sign also has the advantage that it could easily be replicated for different kinds of businesses.

The supported task is the customer experience deciding whether or not to visit the café (Fig. 7.12). It is pretty much identical, equivalent to the chip-van customer task (Fig. 7.10).

Rory Gianni created the Internet-enabled open sign (probably the world's first), using an Electric Imp (see Fig. 7.13). The Electric Imp contains a Wi-fi and cloud enabled processor in a package rather like a large SD card. This has been specifically designed for Internet of Things applications and can be connected to various sensors and actuators. It is programmed and runs code through Electric Imp's cloud platform. For this purpose, no explicit sensor was needed as the power for the device was simply connected to the open sign's low-voltage power input, meaning it is only on when the open sign is on. When it powers up, it sends a periodic message back to the cloud platform, which in turn calls a small script on the Tiree Tech Wave site where a

Fig. 7.11 Cobbled Cow café—sensed task (owner)

0. Running cafe

1. drive to cafe
2. enter cafe (through side door)
3. prepare for opening:
   turns on power, lights, etc.
4. open up cafe (at opening time)
   - 4.1 turn on open sign
     ** sensed by system
   - 4.2 open café doors
5. serving customers
   take order, cook and serve food, wrap, take money
6. close up cafe (at closing time)
   - 6.1 close café doors
   - 6.2 turn off open sign
     ** sensed by system
7. tidy up
8. leave cafe (side door)
9. go home

0. Eat out at café

1. decide would like food at café
2. check if open
   - 2.1 check current time and opening hours (old)
   - 2.2 look at open sign on web page (new)
3. drive to café
4. buy and eat food (if open)

   or
5. disappointed and hungry (if closed)
6. drive home

Fig. 7.12 Cobbled Cow café—supported task (customer)

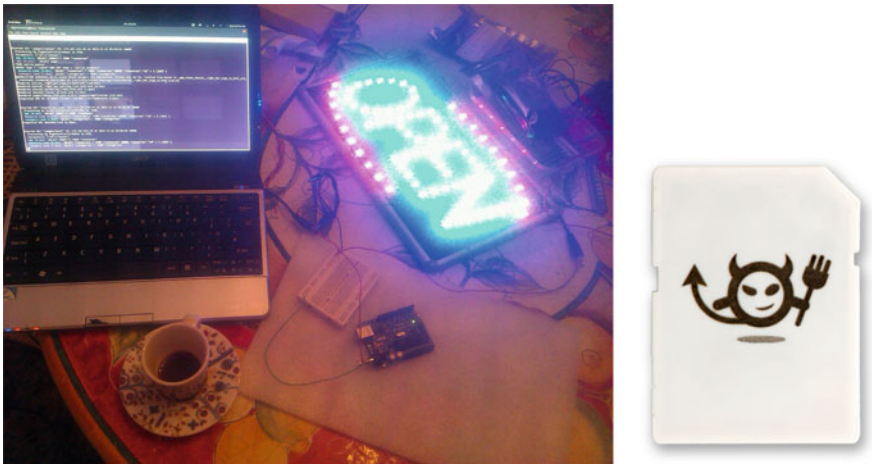

Fig. 7.13 *Left* Internet-enabled open sign under development (photograph Rory Gianni). *Right* Electric Imp module (photograph www.electricimp.com media resources)

log is stored. When the Web-based status page is requested, this checks the logs and can display a Web open sign, which reflects the state of the physical one.

The Internet-enabled open sign was deployed at the Cobbled Cow and has been running now for more than 2 years.

In many ways, the system at the Cobbled Cow resembles the chip van prototype, with a few differences. Technically, rather than Twitter as middleware, the Electric Imp cloud service is being used, and rather than a model chip van on the mantlepiece, a simple Web sign is used for displaying the status. More critically, there are differences in the interaction probabilities.

In the case of the chip van, it is impossible to serve customers without opening the flap. That is for the chip van sensed task (Fig. 7.9), sub-task 4 (open flap) will always happen before sub-task 5 (serve customers). In contrast, for the Cobbled Cow sensed task (Fig. 7.11), sub-task 4.1 (turn on open sign) is a matter of routine, but it is physically possible to open the café (sub-task 5) without turning on the

sign; that is sub-task 4.1 is likely but not certain to occur. Similarly, it is possible to close the shop without turning off the sign (sub-task 6.2), although this is likely to be noticed at sub-task 8 (leave shop) as the glow of the sign would be visible except in high summer.

That is the uncertain nature of tasks being sensed means that there is a possibility that the Web sign might be off in the daytime when the café is actually open, or on at night when it is actually closed. The latter is fairly obvious to the viewer, but the former could mean they do not go to the café when it is in fact open. Arguably, this is not as bad as a customer going when it is actually closed, but still a problem.

Figure 7.14 shows these issues using a diagram of the main states of the café, similar to that for the car courtesy light in Fig. 7.8. In Fig. 7.14, the states where we would like the Internet version of the sign to show 'open' (when the shop is open) are shown with thick edges. The states shown dashed are those where the sign is 'wrong' (on when the shop is closed or vice versa). The transitions labelled with roman numerals (i), (ii), (iii), (iv), (v), and (vi) are the normative path corresponding,

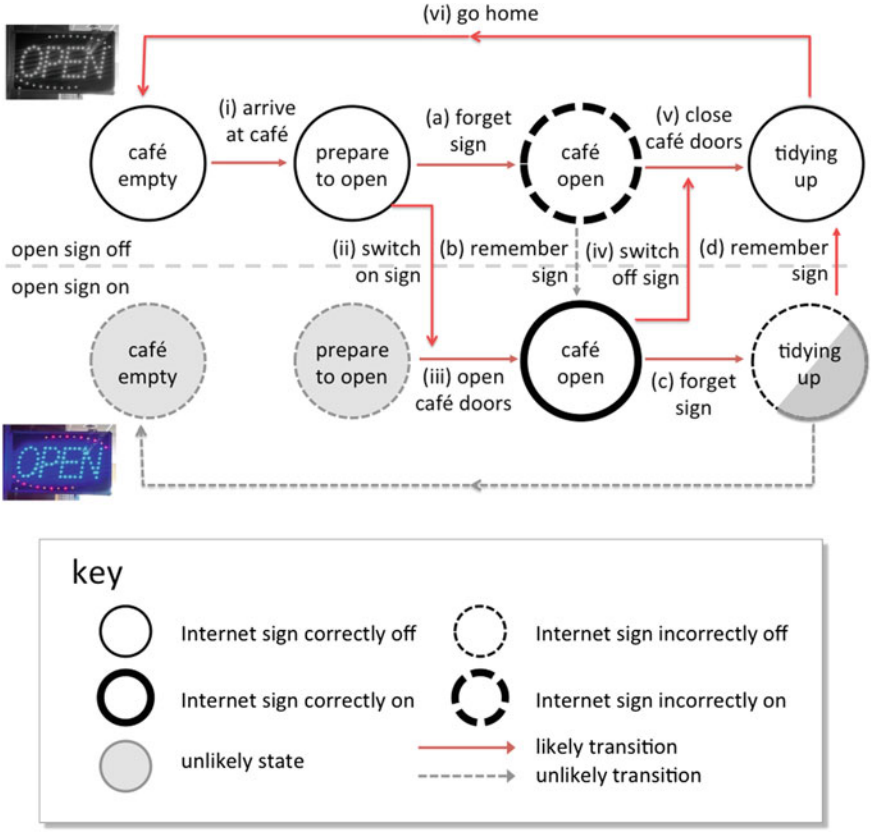

Fig. 7.14 States of the café

Fig. 7.15 Public LED 'ticker-tape' display showing local weather, news, etc

respectively, to steps in Fig. 7.11: 2, 4.1, 4.2, 6.2, 6.1 and 8. So long as the café owners follow these transitions, the sign will always be correct.

However, some transitions lead from good states to bad states: (a) forgetting to turn on the sign when the café is opened; and (c) forgetting to turn it off when closing. In each case, there is a possible correction: (b) noticing the sign is off when the café is open, and (d) noticing it is on when the café is being tidied up at the end of the day. Transition (b) is marked dashed and in grey to signify that it is less likely to occur, because in the daytime it is not easy to see that the sign is not on from inside the café. In contrast, transition (d) is solid to signify it is more likely to occur, as the glow of the sign is pretty obvious as you turn out lights, although sometimes this could be well after the shop is actually closed. Because of the latter, the transition denoting leaving with the sign on is greyed out to signify it is unlikely. Based on the likelihoods of these transitions, some states have been marked in grey to signify that they are unlikely to occur.

As can be seen, the only state that is both wrong and likely to occur is the sign being off when the café is open. This could be addressed by finding some means to make forgetting to turn on the sign (a) less likely. Alternatively, one could try to make it easier to notice when the sign is not on. As an attempt to partly address this, the LED 'ticker-tape' information display (Fig. 7.15) was wired so that both it and the open sign are turned on together. While the open sign faces outwards through the window, the ticker-tape faces inwards into the café. This can still be forgotten, but it is more likely that it will be noticed, that is making transition (b) more likely to occur at all, and transition (d) likely to occur more promptly after the shop has closed.

7.6 Further Design Considerations for Low Intention

The techniques above have been focused on the 'functional' design of low-intention systems, choosing appropriate sensors from the sensed task to allow suitable interventions to the supported task. However, these are far from the only important design issues. We will look at two of these: user models and privacy.

7.6.1 User Models

This chapter is principally concerned with designer models for low-intention interaction, but of course, users also have (or maybe fail to have) models of these systems. This is well explored for more traditional systems with considerable work on metaphors, and design guidelines related to aspects such as visibility, consistency and feedback, which help the user to understand the system states, the available actions and the effects of these on the system.

Considering context-aware systems, Schmidt (2013) suggests that the user interface should seek to minimise 'awareness mismatch', making it clear to users what sensory information is being used, so that the user can make satisfactory explanations of why system effects occurred and valid inferences about how to create desired outcomes.

For some of the best low-intention and low-attention interfaces, none of this is necessary. If users are unaware that modifications are happening, and any alterations in interaction are sufficiently good, then they do not need to build any sort of model. For example, many users are unaware that search results are tuned to their past click-through behaviour.

However, if the user does become aware, for example that the heating or lighting levels spontaneously change, then the interaction may become 'spooky', as if there are ghosts in the walls changing the very environment. Intentional interactions may sometimes be indirect (e.g. waving your arm to control the character on a video game), but are extensions of the physical actions of day-to-day life. Autonomous behaviour, however, often suggests animate beings at work. This is rather like the 'uncanny valley' (Mori 1970) for human-like robots, the more intelligent and human-like autonomous action becomes, the more like magic, or the supernatural it becomes.

This suggests that, on a moment-to-moment basis, low-intention systems need not follow Schmidt's (2013) 'awareness mismatch' advice. However, some sort of model should be available when users want it. The latter also helps address some of the ethical issues of making modifications that are not apparent to users. In Europe, this issue is likely to become increasingly important following the General Data Protection Regulation of the Council of the European Union (2016), which bans or severely regulates black box automatic decision-making in any legally sensitive area (Goodman and Flaxman 2016).

7.6.2 Privacy

Sensor-rich environments also raise privacy and security issues. The definition of incidental interaction says sensed data is collected '*in order to influence/improve/facilitate the actors' future interaction or day-to-day life*' (Dix 2002)—that is the individual or group being sensed also personally benefits from

that sensing. This was phrased deliberately to exclude surveillance where data are collected purely for the benefit of others.

Of course, even where benefits do accrue to the person being sensed, this does not mean that their data may not be deliberately or accidentally misused. The usage data that sites such as Google or Twitter collect are used to enhance your own experience, but also to feed marketing databases. Cloud-based systems can serve to make this worse, and there have been a number of recent news articles worrying about smart TVs and voice interaction dolls, which send anything spoken in their vicinity to central servers to be processed and often stored.

There have been attempts to use similar technology to reverse the balance. Steve Mann and colleagues deliberately use personal wearable technology to record as a visitor in environments, such as shops, where CCTV and other means are usually used to record those coming in, a practice they term '*sousveillance*' (Mann et al. 2003).

Of course, this itself may invade the privacy of others and create a new disparity, potentially leading to antagonism against 'cyborgs', perhaps most notably when Mann's 'EyeTap' digital glasses were removed in a Paris McDonald's restaurant (Biggs 2012; Popper 2012). As this kind of technology becomes commoditised, for example with Google Glass or even ubiquitous mobile phone video, both kinds of privacy issue converge.

7.6.3 Can Task Models Help?

None of the above issues are explicitly dealt with by the task modelling suggested in this chapter, but the two-task view does help to elucidate some of the issues and the way they relate to one another.

The user modelling issues are primarily related to the supported task. Clearly identifying the points at which interventions happen in this can at least help the designer to assess the potential for 'spookiness' and also make it easier to create explanation systems. The uncertainty related to sensors can of course make this worse as system actions are less predictable, for example if your phone battery dies and so the heating system adjusts the house based on the wrong inferred prior activity.

In contrast, privacy issues are primarily related to the sensed task. While such issues are not 'solved' by task modelling, having a clear specification of where and when data are gathered makes it easier both to avoid unintended disclosure and to explain to users what is being gathered and why, thus ensuring informed consent.

Both issues are made more problematic by complex inference algorithms used as either part of sensor fusion or context-sensitive interactions.

Privacy frameworks in the ubicomp literature focus on *restricting* information flows, in the belief that less information means more privacy. However, the earliest work on privacy in HCI showed that on occasions, less information, even highly

anonymised information, could be more personally sensitive than full information (Dix 1990); indeed, half-stories are the grist of gossip and the downfall of many a politician.

Just as problematic are actions based on black box algorithms. Again, the author's early work on the HCI implications of pattern matching technology highlighted the danger that algorithms could unintentionally produce sexist or racist outputs (Dix 1992). This issue has taken some time to become apparent, but has now come to public attention with high-profile reporting of apparently racist search results (Gibbs 2015) and photo tagging (BBC 2015; Hern 2015). More mundane examples such as intelligent heating might seem immune from such issues, except that it has been recently argued that office workplace temperature standards are effectively sexist, based on typical male metabolism and hence disadvantaging women (Kingma and van Marken Lichtenbelt 2015). Because the standards are public, this can be exposed, challenged and debated. Quite possibly, a more intelligent office air-conditioning system might avoid this problem, basing temperature on actual occupancy, but equally it could make things worse by introducing new implicit and potentially illegal bias. Most worrying, it would be far harder to tell whether it was discriminatory or, even if it is not, defend against the accusation that it is.

In both cases, provenance of data and perspicuity of algorithms are at the heart of helping to ensure that users of these systems can make sense of the outcomes. The two-task analysis helps in the former, because it factors the gathering of data from the application of that data; but it does not solve the problem of inscrutable algorithms.

7.7 Discussion

We have seen how it is possible to analyse the twin tasks for low-intention interaction to identify the potential, albeit uncertain, sensor data from the sensed task, in order to modify system behaviour for the supported task. In the first example, the car courtesy lights, the two tasks were effectively the same activity, whereas in the second example, the Internet-enabled open sign, the sensed and supported tasks were different.

Critical in both examples has been the assessment of the likelihood that certain sensor measurements will indicate particular user behaviour. In some cases, this can be clear-cut: if the car has started and is moving, then it is occupied. In others, it is not definitive, merely indicative: in the car example, opening the doors may mean a person is getting in the car; when the sign is powered on, it may indicate that the shop is open; but in both cases, there are circumstances when the sensor and user behaviour may not match.

Note also that in some cases there are physical processes at work that constrain user behaviour (e.g. you cannot get into a car normally without opening a door), but in others, we rely on habit, routine or 'typical' user behaviour (e.g. the café owner's opening up rituals).

To some extent, the methods and techniques described in this chapter have become 'second nature' to the author, and so during the design process for TireeOpen, the formal models were mostly 'in the head' rather than on paper; however, the steps described in Sect. 6 accurately reflect the processes and analyses used.

This chapter has not presented a specific notation, but instead has shown how standard notations for describing tasks, scenarios and physical models can be annotated and analysed in order to aid the design of low-intention systems. Also, it has not attempted to connect this design modelling to the more implementation-oriented context-modelling notations. Doing this could enable a level of verification of the implemented system with respect to the design intentions.

As noted, low-intention systems are already ubiquitous in the digital domain, and we encounter mundane examples in day-to-day life. However, as the Internet of Things and ubiquitous computing move from vision to reality, we will live in an increasingly digitally augmented physical environment. The techniques in this chapter are one step in addressing some of the issues that arise as we seek to design systems for this emerging world.

Acknowledgements Many thanks to Rory Gianni who created the Internet-enabled open sign.

References

Alexander A, Ishikawa S, Silverstein M Ingrid K, Angel S, Jacobsen M (1977) A pattern language. towns, buildings, construction. Oxford University Press. https://archive.org/details/APatternLanguage

BBC (2015) Google apologises for Photos app's racist blunder. BBC News, Technology. http://www.bbc.co.uk/news/technology-33347866. Accessed 1 July 2015

Beigl M, Gellersen H, Schmidt A (2001) MediaCups: experience with design and use of computer-augmented everyday objects. Comput Netw 35(4):401–409

Benford S, Schnadelbach H, Koleva B, Gaver B, Schmidt A, Boucher A, Steed A, Anastasi R, Greenhalgh C, Rodden T, Gellersen H (2005) Expected, sensed, and desired: a framework for designing sensing-based interaction. ACM Trans Comput-Hum Interact (TOCHI) 12(1):3–30

Biggs J (2012) Augmented reality explorer Steve Mann assaulted at Parisian McDonald's. TechCrunch. http://techcrunch.com/2012/07/16/augmented-reality-explorer-steve-mann-assaulted-at-parisian-mcdonalds/. Accessed 16 July 2012

Boer L, Donovan J (2012) Provotypes for participatory innovation. In: Proceedings of the DIS '12. ACM, pp 388–397. doi:10.1145/2317956.2318014

Bruegger P, Lisowska A, Lalanne D, Hirsbrunner B (2010) Enriching the design and prototyping loop: a set of tools to support the creation of activity-based pervasive applications. J Mobile Multim 6(4):339–360

Bruegger P (2011) uMove: a wholistic framework to design and implement ubiquitous computing systems supporting user's activity and situation. PhD thesis, University of Fribourg, Switzerland. http://doc.rero.ch/record/24442

Council of the European Union (2016) Position of the council on general data protection regulation. http://www.europarl.europa.eu/sed/doc/news/document/CONS_CONS(2016)05418(REV1)_EN.docx. Accessed 8 April 2016

Dey A (2001) Understanding and using context. Pers Ubiquit Comput J 5(1):4–7
Dix A (1990) Information processing, context and privacy. In: Diaper G, Cockton G, Shakel B (eds) Human-computer interaction—INTERACT'90 North-Holland. pp 15–20. http://alandix.com/academic/papers/int90/
Dix A (1991) Formal methods for interactive systems. Academic Press, London. ISBN 0-12-218315-0. http://www.hiraeth.com/books/formal/
Dix A (1992) Human issues in the use of pattern recognition techniques. In: Beale R, Finlay J (eds) Neural networks and pattern recognition in human computer interaction, Ellis Horwood, pp 429–451. http://alandix.com/academic/papers/neuro92/neuro92.html
Dix A, Abowd G (1996) Modelling status and event behaviour of interactive systems. Softw Eng J 11(6):334–346
Dix A, Rodden T, Davies N, Trevor J, Friday A, Palfreyman K (2000a) Exploiting space and location as a design framework for interactive mobile systems. ACM Trans Comput-Hum Interact (TOCHI) 7(3):285–321
Dix A, Beale R, Wood A (2000b) Architectures to make simple visualisations using simple systems. In: Proceedings of the working conference on advanced visual interfaces (AVI '00). ACM, New York, USA, pp 51–60. doi:http://dx.doi.org/10.1145/345513.345250
Dix A (2002) Beyond Intention—pushing boundaries with incidental interaction. In: Proceedings of building bridges: interdisciplinary context-sensitive computing. Glasgow University. http://alandix.com/academic/papers/beyond-intention-2002/. Accessed 9 Sept 2002
Dix A, Finlay J, Abowd G, Beale R (2004) Modeling rich interaction. Chapter 18 in Human–computer interaction, 3rd edn. Prentice Hall, Englewood Cliffs. http://www.hcibook.com/e3/
Dix A (2006) Car courtesy lights—designing incidental interaction. Case study in HC Book online! 2004/2006. http://www.hcibook.com/e3/casestudy/car-lights/
Dix A, Leite J, Friday A (2007). XSED—XML-based description of status-event components and systems. In: Proceedings of engineering interactive systems 2007, IFIP WG2.7/13.4 conference, Salamanca, March 22–24, 2007, LNCS, vol 4940, pp 210–226
Dostal J, Dix A (2011) Tiree tech wave. Interfaces, Summer 2011, pp 16–17 http://tireetechwave.org/events/ttw-1/interfaces-article/
Forbrig P, Märtin C, Zaki M (2013) Special challenges for models and patterns in smart environments. In: Kurosu M (ed) Proceedings of the 15th international conference on human-computer interaction: human-centred design approaches, methods, tools, and environments—Volume Part I (HCI'13), vol Part I. Springer, Berlin, pp 340–349
Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object-oriented software. Addison-Wesley, Longman
Gellersen H, Beigl M, Krull H (1999) The MediaCup: awareness technology embedded in an everyday object. In: International symposium on handheld and ubiquitous computing (HUC99), Karlsruhe, Germany
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 conference on design computing & cognition 2006, TU/e Eindhoven. http://alandix.com/academic/papers/DCC-2006-LGP-natural-inverse/. Accessed 8–12 July 2006
Gibbs S (2015) Google says sorry over racist Google Maps White House search results. The Guardian, 20 May 2015. http://www.theguardian.com/technology/2015/may/20/google-apologises-racist-google-maps-white-house-search-results
Goodman B, Flaxman S (2016) EU regulations on algorithmic decision-making and a "right to explanation". Presented at 2016 ICML workshop on human interpretability in machine learning (WHI 2016), New York, NY. http://arxiv.org/abs/1606.08813v1
Green T, Petre M M (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. J Visual Lang Comput 7(2):131–174. doi:10.1006/jvlc.1996.0009
Greenfield A (2006) Everyware: the dawning age of ubiquitous computing. Peachpit Press, Berkeley
Heidegger M (1927) Sein und Zeit. (English translation: Being and Time. Harper, 2008)

Hern A (2015) Flickr faces complaints over 'offensive' auto-tagging for photos. The Guardian. http://www.theguardian.com/technology/2015/may/20/flickr-complaints-offensive-auto-tagging-photos. Accessed 20 May 2015

Hinze A, Malik P, Malik R (2006) Interaction design for a mobile context-aware system using discrete event modelling. In: Estivill-Castro V, Dobbie G (eds) Proceedings of the 29th Australasian computer science conference, vol 48 (ACSC '06). Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp 257–266

Hutchins E, Holland J, Norman D (1986) Direct manipulation interfaces. In: Norman DA, Draper SW (eds) User centered system design. Lawrence Erlbaum Associates, Hillsdale, NJ, pp 87–124

Hutchinson H, Mackay W, Westerlund B, Bederson B, Druin A, Plaisant C, Beaudouin-Lafon M. Conversy S, Evans H, Hansen H, Roussel N, Eiderbäck B (2003) Technology probes: inspiring design for and with families. In: CHI'03: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 17–24

Ju W, Leifer L (2008) The design of implicit interactions: making interactive systems less obnoxious. Des Issues 24(3):72–84. doi:10.1162/desi.2008.24.3.72

Kingma B, van Marken Lichtenbelt W (2015) Energy consumption in buildings and female thermal demand. Nat Clim Change 5:1054–1056. doi:10.1038/nclimate2741

Mann S, Nolan J, Wellman B (2003) Sousveillance: inventing and using wearable computing devices for data collection in surveillance environments. Surv Soc 1(3):331–355. http://www.surveillance-and-society.org/articles1(3)/sousveillance.pdf

Mori M (1970). The uncanny valley. Energy 7(4):33–35. (in Japanese). Translated MacDorman K, Kageki N (2012) IEEE Spectrum. http://spectrum.ieee.org/automaton/robotics/humanoids/the-uncanny-valley. Accessed 12 Jun 2012

Newman W, Eldridge M, Lamming M (1991) Pepys: generating autobiographies by automatic tracking. In: Proceedings of the second European conference on computer supported cooperative work—ECSCW '91, 25–27 Sept 1991, Kluwer Academic Publishers, Amsterdam, pp 175–188

Nielsen J (1993) Usability engineering. Academic Press, San Diego

Norman D (1990) The design of everyday things. Doubleday, New York

Norman D (2010) Natural user interfaces are not natural. Interactions 17(3):6–10. doi:10.1145/1744161.1744163

Pallotta V, Bruegger P, Hirsbrunner B (2008) Kinetic user interfaces: physical embodied interaction with mobile ubiquitous computing systems. In: Kouadri-Mostéfaoui S, Maamar M, Giaglis P (eds) Advances in Ubiquitous computing: future paradigms and directions. IGI Global Publishing, pp 201–228. ISBN:978-1-599-04840-6

Paterno F (2012) Model-based design and evaluation of interactive applications. Springer

Popper B (2012) New evidence emerges in alleged assault on cyborg at Paris McDonald's. The Verge. http://www.theverge.com/2012/7/19/3169889/steve-mann-cyborg-assault-mcdonalds-eyetap-paris. Accessed 19 July 2012

Rehman K, Stajano F, Coulouris G (2007) An architecture for interactive context-aware applications. IEEE Perv Comput 6(1):73–80. doi:http://dx.doi.org/10.1109/MPRV.2007.5

Salber D, Dey A, Abowd G (1999) The context toolkit: aiding the development of context-enabled applications. In: Proceedings of the 1999 conference on human factors in computing systems (CHI '99), Pittsburgh, PA, 15–20 May 1999, pp 434–441

Schmidt A (2013) Context-aware computing. The encyclopedia of human–computer interaction, 2nd edn. In: Soegaard M, Friis R (eds) Interaction design foundation. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/context-aware-computing-context-awareness-context-aware-user-interfaces-and-implicit-interaction

Schmidt A (2000) Implicit human computer interaction through context. Pers Technol 4(2–3):191–199. doi:10.1007/BF01324126

Schilit B, Adams N, Want R (1994) Context-aware computing applications. In: Proceedings of the 1994 first workshop on mobile computing systems and applications (WMCSA '94). IEEE

Computer Society, Washington, DC, USA, pp 85–90. doi:http://dx.doi.org/10.1109/WMCSA.1994.16

Shepherd A (1989) Analysis and training in information technology tasks. In: Diaper D (ed) Task analysis for human-computer interaction, Chapter 1. Ellis Horwood, Chichester, pp 15–55

Shneiderman B (1982) The future of interactive systems and the emergence of direct manipulation. Behav Inf Technol 1(3):237–256

Shneiderman B (1998) Designing the user interface—Strategies for effective human-computer interaction, 3rd edn. Addison Wesley

Tang L, Yu Z, Wang H, Zhou X, Duan Z (2014) Methodology and tools for pervasive application development. Int J Distrib Sens Netw 10(4). http://dx.doi.org/10.1155/2014/516432

Want R, Schilit B, Adams N, Gold R, Petersen K, Goldberg D, Ellis J, Weiser M (1995) An overview of the PARCTAB ubiquitous computing experiment. IEEE Pers Commun 2(6):28–43. doi:10.1109/98.475986

Wigdor D, Wixon D (2011) Brave NUI world: designing natural user interfaces for touch and gesture. Morgan Kaufmann

Wilde A, Pascal B, Hirsbrunner B (2010) An overview of human-computer interaction patterns in pervasive systems. In: International conference on user science and engineering 2010 (i-USEr 2010), Sha-Allam, Malaysia, 13–15 Dec 2010

Wurdel M (2011) An integrated formal task specification method for smart environments. University of Rostock. ISBN:978-3-8325-2948-2