Considering Context and Users in Interactive Systems Analysis

José Creissac Campos[1], Michael D. Harrison[2]

[1] DI/CCTC, Universidade do Minho
Campus de Gualtar, 4710-057 Braga, Portugal
Jose.Campos@di.uminho.pt

[2] Informatics Research Institute, Newcastle University
Newcastle upon Tyne, NE1 7RU, UK
Michael.Harrison@ncl.ac.uk

Abstract. Although the take-up of formal approaches to modelling and reasoning about software has been slow, there has been recent interest and facility in the use of automated reasoning techniques such as model checking [5] on increasingly complex systems. In the case of interactive systems, formal methods can be particularly useful in reasoning about systems that involve complex interactions. These techniques for the analysis of interactive systems typically focus on the device and leave the context of use undocumented. In this paper we look at models that incorporate complexity explicitly, and discuss how they can be used in a formal setting. The paper is concerned particularly with the type of analysis that can be performed with them.

Keywords: interactive systems, modelling, analysis, context.

1 Introduction

Because usability is dependent on "specified users" [11], at the limit the usability of a device can only be assessed empirically and 'in situ'. However, usability analysis techniques can be employed to help the designers and developers to envisage the impact of interactive systems.

Different types of usability analysis methods have been proposed over the years. They can be divided into two general classes. Empirical methods (typically performed with real users – for example, think aloud protocols and questionnaires), and analytic models (usually based on models – for example, heuristic evaluation and cognitive walkthroughs).

Usability cannot be guaranteed in an analytic way. There are simply too many factors involved to make it feasible. Nevertheless, despite some dispute about their real worth [9, 10], analytic methods are being used in practice and evidence indicates that they can play a relevant role in detecting potential usability problems from the outset of design [6].

Performing usability analysis of interactive systems design is a multi-faceted problem. This means that no single analysis method can cover all aspects of usability. For example, Cognitive Walkthrough [12] focuses on how the device supports the users' work, while Heuristic Evaluation [13] focuses on generic/universal properties of the device. Different methods will be needed at different stages of design and for different tasks.

One specific type of analytic approach is the use of formal (mathematically rigorous) methods of modelling and reasoning. Although take up of formal approaches to modelling and reasoning about software has been slow, recent years have seen an increased interest in the use of automated reasoning techniques such as model checking [5] for the analysis of complex systems. In the case of interactive systems, formal methods can be particularly useful in reasoning about systems with complex interactions. Examples include the analysis of the internal mode structure of devices [4, 8] and the analysis of the menu structures of interactive applications [19].

Consider, for example, performing a Cognitive Walkthrough of a user interface with a complex mode structure. It will be very difficult, if not impossible, to guarantee that all possible systems response will have been considered during the analysis. With model checking, although we cannot achieve the same level of reasoning about cognitive psychology aspects of the interaction, we are able to test properties over all possible behaviours of the system.

The problem with all these techniques is that they focus on the device, occasionally (as in the case of Cognitive Walkthrough) a representation of the user's task, but never on an explicit representation of the context in which the device and user are embedded. Although in practice the analyst or team of analysts brings this contextual understanding to the table, as devices become more dependent on context the need to make assumptions explicit about context becomes more important. This problem becomes more pressing as we move towards ubiquitous computing where device action uses context explicitly, including details like location, user preferences and previous activity.

In this paper we look at the modelling of interactive systems in a formal setting, and what type of analysis can be performed with them. In particular, we look at how we can consider context in interactive systems modelling and analysis from a formal (mathematically rigorous) standpoint. The contribution of the paper is to develop a separable model of context that supports clarity of assumptions in the analysis of the device.

The structure of the paper is as follows. Section 2 discusses the relevance of user and context considerations in the modelling and analysis of interactive systems. Section 3 addresses modelling of devices. Section 4 addresses modelling of assumptions about user behaviour as restrictions on the behaviour of the device. Section 4 addresses the impact of context in the analysis. Section 5 reflects on what has been presented in the paper. Section 6 concludes with some final considerations.

2 Devices and Users in Context

According to the ISO 9241-11 standard, usability can be defined as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [11]. Analysing this definition, we can see that the factors that have an impact on the usability of a system when trying to achieve a given goal are the actual product (or device) being used, the users using the device to achieve the goal, and the context of the interactive system. From now on we will use the terms *interactive device* (or simply *device*) and *user(s)* to refer to the interactive product being designed/analysed, and to the human(s) using it, respectively. The term *interactive system* will be used to refer to the combination of both (device and users).

Traditionally, analytic approaches to usability analysis have placed particular emphasis on the device and/or user. So, for example, in heuristic evaluation a team of experts checks a model/prototype against a list of desirable features of interactive devices. It is assumed that the experts will identify appropriate usage considerations. Cognitive walkthroughs attempt to determine if/how a device will support its users in achieving specified goals, from a model of the device. The approach is rooted in the CE+ theory of exploratory learning [16], and, in some ways, this means it over prescribes the assumptions that are made about how the user will behave. In PUMA [1] the model of a (rational) user is built to analyze what the user must know to successfully interact with the device. Again, this means that the assumptions about user behaviour are quite strong. In [4] a model of the device is analysed against all possible user behaviour. Instead of prescribing, from the outset, assumptions about how users will behave, these assumptions are derived during the analysis process. Hence, assumptions about the user are identified that are needed to guarantee specified properties of the overall interactive system.

In summary, context has not been given particular attention, being usually only implicitly considered. Taking account of context is important because it has an effect on the way device actions are interpreted. A key problem associated with ubiquitous systems is that confusions arise because actions are interpreted through implicit assumptions about context. This problem is effectively the mode problem that model checking techniques are particularly well suited to addressing.

Additionally, considerations about the user tend to be either too vague (c.f. Heuristic Evaluation) or over prescribed and therefore in danger of not capturing all relevant behaviours (c.f. Cognitive Walkthroughs or PUMA) – these techniques might overlook workarounds for example. While these approaches can be useful, problems arise when we consider complex systems. This happens because it becomes difficult to identify informally all the assumptions that are being made about (or, more importantly, are relevant to) the user behaviour, and/or because a very prescriptive model of user behaviour might rule out unexpected behaviours that are potentially interesting from an analysis point of view.

As stated above, in this paper we are specifically interested in (formal) analytic approaches. We are particularly interested in seeing how we can build on the work developed in [4, 2] to take into consideration models/assumptions about the users and the context of usage of the systems.

In order to make the discussion more concrete, we will be using as a basis an example described in [4] (but considerably reworked here due to our new focus). We need to be clear about what we mean by context. So we want to discuss the issues associated with context using a very simple example. Rather than look at a ubiquitous system we re-consider the analysis of a mode control panel (MCP). This is a safety critical interactive system that has been analysed using a number of techniques [14]. The important thing about this example is that the context in which the device is embedded is crucial to an understanding of the interactive behaviour of the system. The techniques that are developed here are as important in intelligent and mobile systems where action inference (qua mode) is based on preferences, or location, or history or other elements that can be described as context. The example addresses the design of the Mode Control Panel (MCP) of an MD-88 aircraft (see figure 1), and was developed using MAL interactors [4, 2].

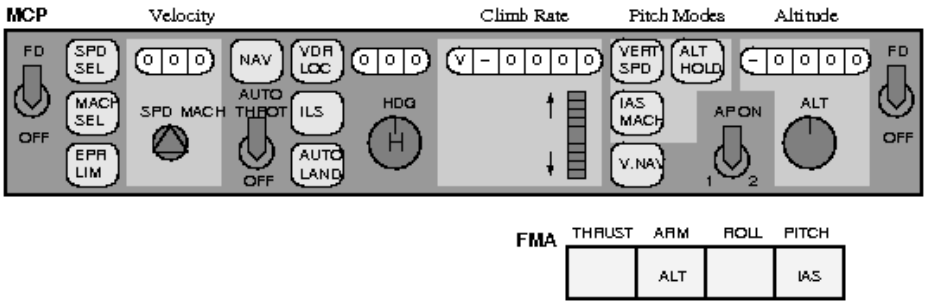

Fig. 1. The MCP panel (areas with lighter background will be modelled)

3 Device Model (or, Devices in Context)

Building a behavioural model of the device enables analysis of all the behaviours that are possible to achieve goals. Whether or not these behaviours are cognitively plausible, however, is sometimes left outside the formal analysis process. This is due to the difficulty in adequately formalising the users' cognitive process. This aspect will be further explored in section 2.2. For now we will concentrate on the device model.

3.1 Modelling

In the approach put forward in [4, 2] only the device is modelled explicitly. In the MCP example, the device is the actual MCP. Using MAL interactors we can perform a first modelling approach[1]:

[1] For brevity the definitions of some named expressions are not presented here. It is expected that the names used will be self-explanatory. The full model is available at http://www.di.uminho.pt/ivy/index.php?downloads

```
interactor MCP
 includes
  dial(ClimbRate) via crDial
  dial(Velocity) via asDial
  dial(Altitude) via ALTDial
 attributes
  [vis] pitchMode: PitchModes
  [vis] ALT: boolean
 actions
  [vis] enterVS enterIAS enterAH enterAC
  toggleALT
 axioms
  [asDial.set(t)] action'=enterIAS
  [crDial.set(t)] action'=enterVS
  [ALTDial.set(t)] ensure_ALT_is_set
  [enterVS] pitchMode'=VERT_SPD & ALT'=ALT
  [enterIAS] pitchMode'=IAS & ALT'=ALT
  [enterAH] pitchMode'=ALT_HLD & ALT'=ALT
  [toggleALT] pitchMode'=pitchMode & ALT'=!ALT
  [enterAC] pitchMode'=ALT_CAP & !ALT'
```

For a description of the MAL interactors language the reader is directed to [2]. Here the focus is not so much on the particular language being used but in what is being expressed. We will provide enough detail about the models to make their meaning clear. The main point about the language is to know that axioms are written in Modal Action Logic [18].

Returning to the model above, it includes the three dials of interest identified in figure 1, as well as attributes to model the pitch mode and the altitude capture switch (ALT). The pitch mode defines how the MCP influences the aircraft:

- VERT_SPD (vertical speed pitch mode) – instructs the aircraft to maintain the climb rate set in the MCP;
- IAS (indicated air speed pitch mode) – instructs the aircraft to maintain the velocity set in the MCP;
- ALT_HLD (altitude hold pitch mode) – instructs the aircraft to maintain the current altitude;
- ALT_CAP (altitude capture pitch mode) – internal mode used to perform a smooth transition from VERT_SPD or IAS to ALT_HLD.

The altitude capture switch, when armed, causes the aircraft to stop climbing when the altitude indicated in the MCP is reached. The available actions are related to selecting the different pitch modes, and setting the values in the dials.

This particular model, however, is of limited interest from a behavioural analysis point of view since it does not consider the semantics of the controlled process. In fact only the logic of the user interface has been modelled. In principle, this can enable us to analyse what are the possible behaviours in the interface. In this case, however, in order for the MCP to have realistic behaviour, we must include in the model information about the process that the MCP is controlling and its constraints (i.e., its context of execution). At the minimum we need to know what the possible responses (behav-

iours) of the process are. Without that we will not be able to analyse the joint behaviour of device and user (the interactive system).

In this case, the context is a very simple model of the aircraft and its position in airspace:

```
interactor airplane
 attributes
  altitude: Altitude
  climbRate: ClimbRate
  airSpeed: Velocity
  thrust: Thrust
 actions
  fly
 axioms
# Process behaviour
  [fly] (altitude'>=altitude-1 & altitude'<=altitude+1)
      & (altitude'<altitude -> climbRate'<0)
      & (altitude'=altitude -> climbRate'=0)
      & (altitude'>altitude -> climbRate'>0)
      & (airSpeed'>=airSpeed-1 & airSpeed'<=airSpeed+1)
      & (airSpeed'<airSpeed -> thrust'<0)
      & (airSpeed'=airSpeed -> thrust'=0)
      & (airSpeed'>airSpeed -> thrust'>0)
# not enough airspeed means the plane falls/stalls
  (airSpeed<minSafeVelocity & altitude>0)->climbRate<0
```

This description is bound to the device model through a number of declarations as described below. Firstly, we must bind the two models architecturally. We do this by inclusion in the MCP of:

```
 includes
  airplane via plane
```

Secondly, creating a behavioural binding requires that the following axioms must be included in the MCP:

```
  per(enterAC) -> (ALT & nearAltitude)
  (ALT & pitchMode!=ALT_CAP & nearAltitude)
                                        -> obl(enterAC)
  pitchMode=VERT_SPD -> plane.climbRate=crDial.needle
  pitchMode=IAS -> plane.airSpeed=asDial.needle
  pitchMode=ALT_HLD -> plane.climbRate=0
  pitchMode=ALT_CAP -> plane.climbRate=1
  (pitchMode=ALT_CAP & plane.altitude=ALTDial.needle)
                                        -> obl(enterAH)
```

What these axioms state is how the process and the device are related. The first two axioms state that action enterAC must be performed when the ALT capture is armed and the aircraft is near enough the target altitude, and that only in those conditions can it be performed. The next four axioms state how the different pitch modes in the de-

vice affect the process. The last axiom states that action enterAH must happen when the target altitude is finally reached.

3.2 Analysis

We can now start testing the device. We will be focussing on detecting potential problems with one of the main functions of the MCP: controlling the altitude acquisition procedure. A reasonable assumption is to consider that, whenever the altitude capture is armed, the aircraft will reach the desired altitude (that is, the altitude set in ALTDial). This assumption can be expressed as:

```
AG((plane.altitude!=ALTDial.needle & ALT)
   ->
   AF(pitchMode=ALT_HLD
      & plane.altitude=ALTDial.needle))
```

What the formula expresses is that whenever the plane is not at the altitude set in the ALTDial, and the ALT capture is armed, then eventually the plane will be at the desired altitude and the pitch mode will be altitude hold (ALT_HLD).

A modelling and verification environment (IVY) that is under development[2] has facilitated the analysis of these models using the SMV model checker [5][3]. With the help of the IVY tool, it is possible to determine that the property above does not hold. The counterexample, produced by NuSMV, shows that the pilot can simply toggle the altitude capture off (see figure 2)[4].

We can conclude that, in order to guarantee the property, we must at least assume a user that will not toggle the altitude capture off. This is a reasonable expectation on the user behaviour which can be expressed without adding to the model by changing the property to consider only those behaviours where the pilot does not disarm the altitude capture:

```
AG((plane.altitude!=ALTDial.needle & ALT)
   ->
   AF((pitchMode=ALT_HLD
       & plane.altitude=ALTDial.needle)
      | action=toggleALT))
```

[2] See http://www.di.uminho.pt/ivy.

[3] To be precise, two versions of SMV are currently being maintained and developed: Cadence SMV, by Cadence labs, and NuSMV. In the current context are using NuSMV.

[4] We present here a graphical representation of the traces produced by NuSMV. This representation is shown at the level of abstraction of the MAL interactors model (hence the presence of actions associated with state transitions). Each column represents the behaviour of a single interactor (except for the first column which acts as a global index to the states produced by the model checker). States (represented by rectangles) can be annotated with information on their attributes (not in this particular case) and/or markers identifying specific state properties. Transitions are labeled with the action that triggers them. The trace representations in this paper have been produced by the trace visualizer component of the IVY tool.

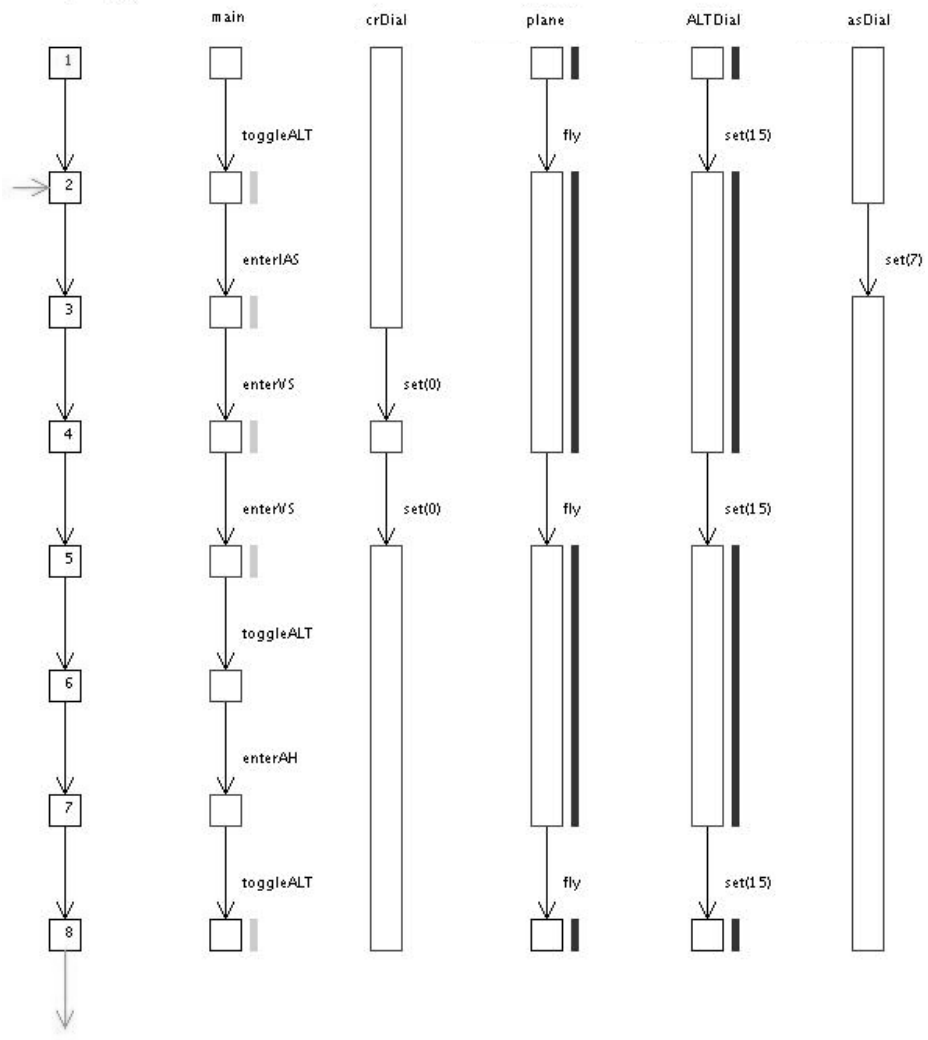

Fig. 2. Counter example for the first property (the dark coloured lines identify states where plane.altitude<ALTDial.needle; the light coloured lines identify states where the ALT capture is armed)

Now, either the plane reaches the desired altitude/pitch mode or the altitude capture is turned off.

This new formulation of the property still does not hold. The counterexample now shows a pilot that keeps adjusting vertical speed. Clearly this is a possible but, in the current context, unlikely behaviour. Once again we need to redefine the property in order to consider only those behaviours where this situation does not happen. There is a limit to the extent to which this process can continue because:

- the property to prove is made opaque through more and more assumptions about the user;

- there are assumptions that can become very hard to encode this way;
- there is no clear separation between the property that was proved and the assumptions that were needed.

To avoid these problems, we will now explore encoding the assumptions about user behaviour as constraints on the possible user behaviours. Remember that up to now we were considering all possible behaviours that the device supported, regardless of their cognitive plausibility. This new approach will be dealt with in the next section.

4 On User and other user related Models

Several authors have proposed the use of different types of models to address the issue of considering users during formal verification of interactive systems. Two examples are the work on Programmable User Modelling Analysis (PUMA) [1], and work by Rushby [17]. In the case of PUMA, the objective is to model a rational user. As already explained, this can become too prescriptive, considering that we want to explore unexpected interactions.

In the case of Rushby's work, assumptions about how the users will behave are encoded in the device model from the outset. The danger here is that no clear separation between the device and user assumptions is enforced by the modelling approach. Hence assumptions might be made that go unnoticed during the analysis.

We adopt an approach similar to the latter except for a significant difference. We do not create the model (make assumptions about user behaviour) beforehand. Instead, we obtain the user model as a bye product of the verification process, identifying the assumptions that are needed for the interactive system to verify the property or properties under consideration. This means that even when the property is finally verified, an analysis must be performed of the needed assumptions in order to see if they are acceptable. This way, the results are less prone to tainting by hidden assumptions made about the users' behaviour during the modelling process.

4.1 Modelling

We will now consider a user model that constrains the pilot not to behave as described in the previous section. The approach to encoding assumptions about user behaviour is to strengthen the pre-conditions on the actions the user might execute.

The only danger in doing this is that the action whose pre-conditions are being strengthened can also be used by the device itself. In that case the axioms would restrict not only user behaviour, but also the device's behaviour. This problem can be avoided by defining distinct user-side, and device-side actions with the same semantics, but different modality annotations.

For example, in the case of the toggleALT action we would be defining two replacement actions:

- toggleALT_user – action for the user to toggle the altitude capture on and off;

- toggleALT_dev – action for the device to toggle the altitude capture on and off.

The first would be marked as user selectable, while the second would not. Alternatively we could use a parameter in toggleALT to specify whether the actions were being caused by the user or by the device, and strengthen the axioms for the user only. In this case, however, using different modalities would not be possible since we would only have one action.

In the current case toggleALT is only performed by the users so we do not need to make the above distinction.

We start by setting up the user interactor. It simply creates a binding (by inclusion to the MCP model):

```
interactor user
 includes
  MCP via ui
```

Next we introduce the assumptions as restrictions on user behaviour. Since we want to model restrictions, the axioms take the form of permission axioms over the action of the user:

- Assumption n. 1 – the pilot will not toggle the altitude capture off. The axiom states that the altitude toggle action is only permitted when the altitude capture is off. This restricts the behaviours of interest to those where the user never switches the altitude capture off. Note that this does not interfere with the internal behaviour of the device. The device uses the enterAC action to switch the capture off when approaching the target altitude.

  ```
  per(ui.toggleALT) -> !ui.ALT
  ```

- Assumption n. 2 – the pilot will be wise enough not to set inappropriate climb rates. The three following axioms state that, when the altitude capture is armed, the user will only set climb rates that are appropriate for the goal at hand (negative if the aircraft is above the target altitude; positive if the aircraft is below the target altitude; and zero when the aircraft is at the target altitude).

  ```
  per(ui.crDial.set(-1)) ->
     (!ui.ALT | ui.plane.altitude>ui.ALTDial.needle)
  per(ui.crDial.set(0)) ->
    (!ui.ALT | ui.plane.altitude=ui.ALTDial.needle)
  per(ui.crDial.set(1)) ->
     (!ui.ALT | ui.plane.altitude<ui.ALTDial.needle)
  ```

Our model is now three tiered. At the core there is the context in which the device is embedded and in which the interaction takes place, in this case the aircraft itself. Then there is the device (the MCP). Finally at the top level there is a model of user assumptions.

4.2 Analysis

We can now test the system under these two user assumptions. Considering the user model, the property becomes:

```
AG((ui.plane.altitude!=ui.ALTDial.needle & ui.ALT)
   -> AF(ui.pitchMode=ALT_HLD
         & ui.plane.altitude= ui.ALTDial.needle))
```

In the context of these two assumptions the property still does not hold. This time the counter example points out that, during the intermediate ALT_CAP pitch mode, changes to the vertical speed will cause a change in pitch mode when the altitude capture is no longer armed. This behaviour effectively 'kills the altitude capture': the aircraft will be flying in VERT_SPD pitch mode with the altitude capture disarmed (see state 7 in figure 3).

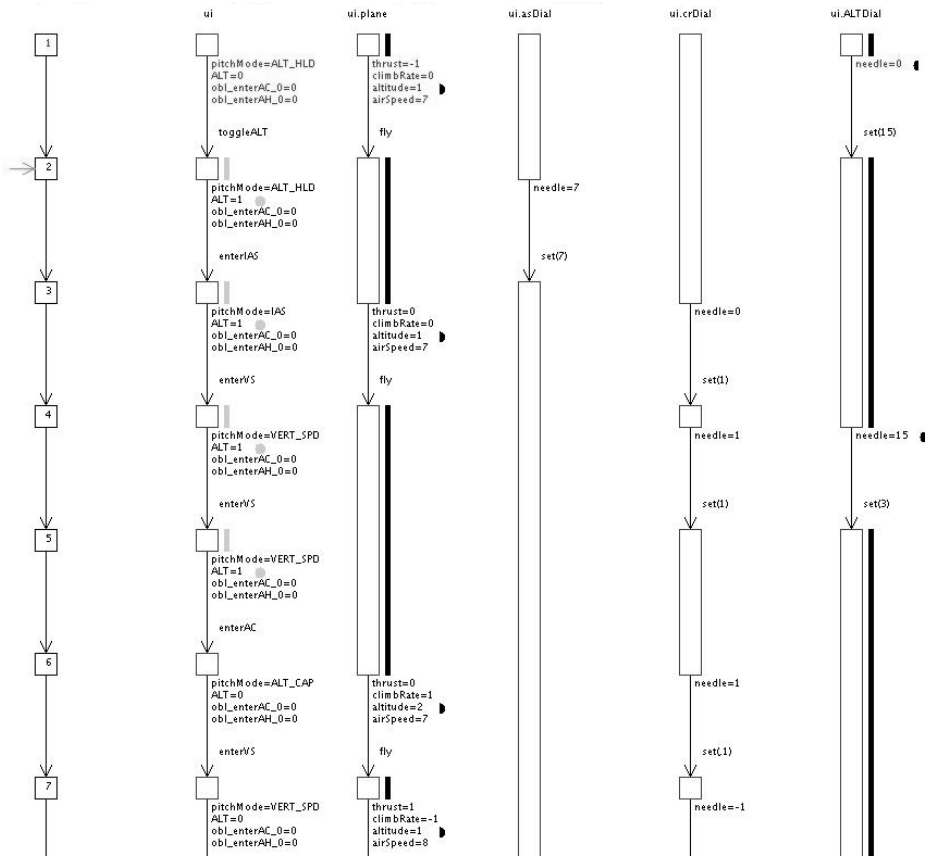

Fig. 3. Partial view of the counter-example for the model with user assumptions (from state 3 to state 4 the action set(1) in crDial causes no problem, from 6 to state 7 the altitude capture is no longer armed and the ALT_CAP pitch mode is lost)

We could keep adding constraints to the behaviour of the user, and we would find out that the only possibility to prove the property is to consider that the user does not make changes to the values set in the MCP while the plane is in ALT_CAP mode. This seems an unreasonable assumption, and in fact instances of this problem have been reported to the Aviation Safety Report System (ASRS) [14].

5 Impact of Context in the Analysis

In reality, there is a problem with the analysis above. We are referring directly to plane.altitude at the user level in the second assumption which is an attribute of the aircraft, not an attribute of the device. On the face of it axioms in the user model should only refer to attributes of the interactive device annotated with an appropriate modality. The problem is that in our model there is no information about current altitude being provided through the device that mediates the context to the user.

There are two possible solutions to this:

- If we are designing the device we might consider including the needed information on the display.
- If we are analysing an existing device (as is the case), or designing it as part of a larger system, we must analyse whether the information is already present in some other part of the system, not included in the current model, and consider how to represent this in the model.

Of course the results of the analysis are completely dependent on the quality of the model used. However, developing separate models for the different levels of analysis involved helps in identifying potential flaws in the models.

In any case, we can also explore the use of contextual information, and whether the needed information is present in the environment.

5.1 Context

Context is understood as the characteristics of the environment that have a bearing on the interactive system (see figure 4). The system (S) is to be understood as the combination of device (D) and user(s) (U). The device is formed by the application's functional core (L) and its user interface (I). Analysing the context can be relevant at a number of levels:

- We might want to analyse whether including some piece of information in the device is really needed – if the information is clearly present in the context of use then including it in the device might create unnecessary user interface clutter.
- We might want to analyse a situation of (partial) system failure, and whether the user will be able to overcome it by resorting to contextual information.
- We might be interested in identifying problems related to different perceptions being obtained from the information gathered through the context and its representation in the device's user interface.

- We might also be interested in the effect that (changes in) the context of usage might have on interaction with the device. It is not the same to use a system under high or low workload conditions. For example, under high workload conditions it is unlikely that the pilot will be able to adequately process the information about vertical speed obtained from the environment.

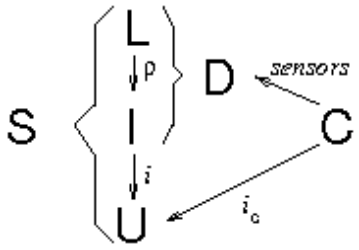

Fig. 4. Context

Context is present at many levels: physical environment, user capability and preferences and so on. Different levels "see" context differently – these may be thought of as interpretation functions (probably partial functions because the levels do not necessarily consider the same subsets of the context, and do not necessarily interpret it in the same way). These different interpretations of context can be used to express how information about context is processed at different levels.

5.2 Context in the MCP

Returning to the MCP, the altitude of the plane is part of the context of the MCP (device). In this case, we can say (assuming a 'large' aircraft) that the pilot has no (or little) context regarding altitude or velocity. He may have information about vertical speed (derived from the aircraft's tilt and thrust). However it is likely that the user perception of this context information is quite low and can be discarded except for extreme circumstances. However, in those extreme circumstances the workload inside the aircraft's cockpit will probably be high. Hence, it is unlikely that the pilot will be able to gain accurate context information. In that case, unless the device provides information on the altitude, the axioms for the first set of assumptions on section 3 cannot be accepted as they have been written.

Even if we consider that contextual information about the altitude is available (because we are talking about a small aircraft), we still have to analyse what information is available. There is the problem of the definition of the information that is perceived by the pilot. It is unlikely that the pilots will be able to compare the altitude displayed in the MCP with their perception of the altitude of the aircraft. It is necessary to be

cautious about what should and should not be part of the context of the user (and how) because this will have a strong impact on the quality of the analysis.

All things considered, it is conservative to assume that the user will not be able to gain accurate enough information regarding altitude from the context of use to be able to compare it with the value set in the ALTDial dial. This means that we must find a way to reformulate assumption number two. As the situation stands, even considering a user that does not use the MCP while in ALT_CAP mode is not enough to conclude that the system is predictable regarding altitude acquisition.

We could simply assume that the pilot would not change the climb rate whenever the altitude capture is armed (or even consider that the MCP would not allow it to happen). These constraints, however, are clearly too strong. The alternative then would be to expand the interface to include information about the current altitude of the aircraft.

We note that while in this case the analysis of contextual information on the user side meant that not enough information was available to users, due to the specific conditions inside a cockpit, in mobile and ubiquitous environments contextual information will most probably play a more relevant role. In this type of system action inference (qua mode) is based on preferences, or location, or history or other elements that can be described as context.

6 Discussion

As stated in section 2, we chose to introduce the issues associated with context by means of a simple example. This was done so that we could be clear about the different concepts involved. This section reflects on what was learnt, and discusses the relevance of context in a larger setting.

6.1 Relevance of context

Figure 4 identifies different aspects that must be considered when analysing an interactive system. The setting of the Activity to be carried out by the system is critical to this analysis. Typical approaches to the analysis of interactive systems that address the interaction between user and interface might or might not take the Activity into consideration (for example, a task model), and might or might not take the Logic of the device into consideration (depending on the modelling detail). What we have argued is that Context is also a relevant factor in this analysis process.

In our example, the aircraft was the context for the MCP and was both being influenced by the MCP, and influencing its behaviour. Hence, context will interact with the device: it can both influence the device's behaviour and be influenced by it.

More importantly, the context will also influence the user. Not only what the user knows (as was discussed in relation to the MCP), but even the user's goals, and how he or she tries to achieve them. Hence, context will also influence the activities the system supports.

6.2 Different models/different analysis

The analysis of the MCP was introduced as a means of illustrating the ideas being put forward regarding both the need to take into account context when performing analysis of interactive system models, and the possibility of deriving information about needed assumptions over user behaviour from that same analysis. It has illustrated a particular style of analysis based on behavioural aspects of the system, specifically related to the mode structure of the device.

Besides mode related issues we can also think of analysing the menu structure of a device, or its support for specific user tasks. Using an approach based on a number of different models, each relating to a specific type of analysis means that it becomes easier to take into consideration different combinations of these factors. For example, we could add to our model a user task model and analyse whether the device, with the given user assumptions, supported that specific task in a given context.

Another (non-mutually exclusive) possibility is to consider the analysis of representational issues of the interface. In fact, it is not sufficient to say that some piece of information is available at the user interface, it is also necessary to consider if the representation being used to present the information is adequate.

Again, the notion of context becomes relevant. In [7] a model of user beliefs about the device's state is analysed against a model of the actual device's state. The objective of that analysis was to assess the quality of the user interface with respect to how it conveyed information about the device. In a contextually rich setting, however, the user will be exposed to more stimuli than those provided by the device, and unless the context of use is considered, the correspondence between the model of user beliefs and reality will be limited.

6.3 Information Resources

Focussing on context not only helps make analysis more accurate by more thoroughly identifying what information users have available, it also raises new issues. Task models might take contextual information into consideration to express how users will adapt to different situations. It becomes relevant to consider how context changes the beliefs the user has about the device, but also how the device conveys information about the context, and whether the information the user receives via the device, and the information the user receives directly are consistent.

The goal of this focus on context is to identify relevant information that the user needs to successfully interact with the system. In the example we were mainly interested in understanding whether the user would have enough information to keep the climb rate of the aircraft at an appropriate level. However, we could also consider what information was needed for the user to take specific actions. For example, if instead of being automatic, the transition to the ALT_CAP pitch mode was to be performed by the pilot, we could be interested in analysing whether enough information was being provided so that the pilot could make the decision to activate that pitch mode at (and only at) the appropriate time.

This information can come from the device or from the context of use. In [3] an approach is discussed that uses the notion of (information) resources to facilitate the

analysis of whether enough information is provided to inform user actions. The resources considered therein related to the device only. The approach can easily be extended to consider contextual information, and to include not only resources for action but also resources as a means of supporting the definition of user assumptions. Hence the notion of information resource can act as a unifying approach that helps in considering all types of information available to the user in the same framework.

7 Conclusion

Several authors have looked at the applicability of automated reasoning tools to interactive systems analysis and their usability characteristics. Approaches such as Paternò's [15] or Thimbleby's [19] have focused heavily on the device. They have shown that it is possible to reason about characteristics of the dialog supported by the device. For example, in [19] it is shown how a formal analysis of the menu structure of a mobile phone could contribute to a simpler and faster dialogue.
When analysing an interactive device, we must take into consideration the characteristics of its users to avoid analysing behaviours that are irrelevant from a cognitive perspective, or consider design that, although ideal according to some formal criterion, are not cognitively adequate. When building a formal model we are necessarily restricting the domain of analysis, and in that process relevant aspects might be left out of the model. This is particularly relevant of interactive systems, where cognitive aspects are important but difficult to capture. Taking the user into consideration during the analysis helps in reducing that effect.

Approaches aimed at building complex architectures that attempt to model the user cognitive processes are clearly inadequate from a verification standpoint. In PUMA [1], a more contained approach is attempted: modelling the behaviour of a rational user. Even so, the authors agree that creating models suitable for automated reasoning is a time consuming process. It should also be noted that the analysis is then performed against those behaviours that are considered rational only. An alternative is to consider, not a model of the user but a model of the work. In [3] information derived from the task model for the device is used to drive the analysis. This enables analysis of whether the device supports the intended tasks, but restricts the analysis to those behaviours that are considered in the task model.

A more flexible approach is to consider assumptions of user behaviour instead of a full blown model of user behaviour or work. These assumptions act as snippets of user behaviour that are found relevant for the analysis in question. Two approaches that follow this approach are work by Campos and Harrison [4] and by Rushby [17]. In the first case assumptions are derived from the analysis process (i.e., nothing is assumed to start with) and the analysis drives which assumptions are needed in order to guarantee some property. The assumptions are encoded into the property under verification. In second approach, assumptions are encoded into the model from the outset. That is, during model development.

The advantage of producing a separate model of context is that (1) it separates the description of the device from those concerns that influence the use of the device (2) it makes clear the contextual assumptions that are being made that can be used as part

of the rationale for the design. Issues of context will become more important with the trend towards ambient systems where user context (for example location, task, history, preferences) may be used by the system to infer what action the user should make.

The example given here hints at many of these issues. This paper sets forth an agenda for more explicit specifications of context that can provide basic assumptions for rationale for the design of implicit action and its analysis.

Acknowledgments. This work was carried out in the context of the IVY project, supported by FCT (the Portuguese Foundation for Science and Technology) and FEDER (the European Regional Development Fund) under contract POSC/EIA/26646/2004.

References

1. Butterworth, R., Blandford, A., Duke, D., Young R.M.: Formal user models and methods for reasoning about interactive behaviour. In J. Siddiqi and C. Roast, editors, Formal Aspects of the Human-Computer Interaction, pages 176–192. SHU Press, 1998.
2. Campos, J.C.: Automated Deduction and Usability Reasoning. DPhil thesis, Department of Computer Science, University of York, September (1999)
3. Campos, J.C., Doherty, G.J.: Supporting resource-based analysis of task information needs. In S. W. Gilroy and M. D. Harrison, editors, Interactive Systems: Design Specification and Verification - 12th International Workshop, DSV-IS 2005, volume 3941 of Lecture Notes in Computer Science, pages 188-200. Springer-Verlag (2006)
4. Campos, J.C., Harrison, M.D.: Model Checking Interactor Specifications. Automated Software Engineering, 8(3):275-310, August (2001)
5. Clarke Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)
6. Desurvire, H.W., Kondziela, J.M., Atwood, M.E.: What is gained and lost when using evaluation methods other than empirical testing. In A. Monk, D. Diaper, and M. D. Harrison, editors, People and Computers VII — Proceedings of HCI' 92, British Computer Society Conference Series, pages 89–102. Cambridge University Press, September (1992)
7. Doherty, G.J., Campos, J.C., Harrison, M.D.: Representational Reasoning and Verification. Formal Aspects of Computing, 12(4):260-277 (2000)
8. Gow, J., Thimbleby, H., Cairns, P.: Automatic Critiques of Interface Modes. In S. W. Gilroy and M. D. Harrison, editors, Interactive Systems: Design Specification and Verification - 12th International Workshop, DSV-IS 2005, volume 3941 of Lecture Notes in Computer Science, pages 201-212. Springer-Verlag (2006)
9. Gray, W., Salzman M.: Damaged merchandise? A review of experiments that compare usability evaluation methods. Human Computer Interaction 13(3):203—261 (1998)
10. Hartson, H.R., Andre, T.S., Williges, R.C.: Criteria for Evaluating Usability Evaluation Methods. International Journal of Human-Computer Interaction, (15)1:145-181 (2003)
11. ISO: International Standard ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on Usability, International Organization for Standardisation, Geneva (1998)
12. Lewis, C., Polson, P., Wharton, C., Rieman, J.: Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In CHI '90 Proceedings, pages 235–242, New York, April. ACM Press (1990)

13. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 249-256. ACM Press (1990)
14. Palmer, E.: "Oops, it didn't arm" – a case study of two automation surprises. In Jensen, R.S. and Rakovan, L.A., eds, Proceedings of the 8th International Symposium on Aviation Psychology', Ohio State University, pp. 227-232 (1995)
15. Paternò, F.D.: A Method for Formal Specification and Verification of Interactive Systems. D.Phil thesis, Department of Computer Science, University of York (1996)
16. Polson, P., Lewis, C.: Theory-Based Design for Easily Learned Interfaces. Human-Computer Interaction, 5:191-220 (1990)
17. Rushby, J.: Using model checking to help discover mode confusions and other automation surprises. Reliability Engineering and Systems Safety, 75(2):167-177, February (2002).
18. Ryan, M., Fiadeiro, J., Maibaum, T.: Sharing actions and attributes in modal action logic. In Ito, T., Meyer, A.R., eds.: Theoretical Aspects of Computer Software. Volume 526 of Lecture Notes in Computer Science, pages 569–593. Springer-Verlag. (1991)
19. Thimbleby, H.: User Interface Design with Matrix Algebra. ACM Transactions on Computer-Human Interaction. 11(2): 181-236. June (2004)