State Event Models for the Formal Analysis of Human-Machine Interactions

Sébastien Combéfis [*], **Dimitra Giannakopoulou** [†] and **Charles Pecheur** [*]
[*]ICT, Electronics and Applied Mathematics Institute
Université catholique de Louvain, Louvain-la-Neuve, Belgium
Email: {sebastien.combefis, charles.pecheur}@uclouvain.be
[†]NASA Ames Research Center
Moffett Field, CA 94035, USA
Email: dimitra.giannakopoulou@nasa.gov

Abstract

The work described in this paper was motivated by our experience with applying a framework for formal analysis of human-machine interactions (HMI) to a realistic model of an autopilot. The framework is built around a formally defined conformance relation called "full-control" between an actual system and the mental model according to which the system is operated. Systems are well-designed if they can be described by relatively simple, full-control, mental models for their human operators. For this reason, our framework supports automated generation of minimal full-control mental models for HMI systems, where both the system and the mental models are described as labelled transition systems (LTS). The autopilot that we analysed has been developed in the NASA Ames HMI prototyping tool ADEPT. In this paper, we describe how we extended the models that our HMI analysis framework handles to allow adequate representation of ADEPT models. We then provide a property-preserving reduction from these extended models to LTSs, to enable application of our LTS-based formal analysis algorithms. Finally, we briefly discuss the analyses we were able to perform on the autopilot model with our extended framework.

Introduction

Despite the fact that the complexity of automated systems constantly increases, it is important to be able to design interfaces that enable human operators to control the system in unambiguous ways. Our work is concerned with providing automated formal analysis techniques for HMI systems. In particular, we have developed a formal framework for the design and analysis of such systems, and have applied it to a number of moderately sized case studies (Combéfis et al. 2011a).

Human operators typically interact with a system based on a conceptual model that they carry in their mind, or that is provided through documentation. In this work, we use the term "mental model" to refer to such conceptual models, not to be confused with cognitive models that are also used in HMI research. Our analysis framework is built around a formally defined conformance relation called "full-control" between an actual system and a mental model according to which it is operated. Systems are well-designed if they can be described by relatively simple, full-control, mental models for their human operators. For this reason, our framework supports automated generation of minimal full-control mental models for HMI systems, where both the system and the mental model are described as labelled transition systems (LTS).

However, in applying our framework to system models developed by HMI collaborators at NASA Ames, we realised that LTSs are conceptually far from the way in which HMI designers typically think about their systems. LTSs are state machines that contain no information on their states except for the transitions and behaviours that are enabled starting from these states. Contrary to that, HMI designers think of their systems in terms of a set of observable and unobservable state variables; different combinations of valuations of these variables constitute the set of possible states of the system, and transitions describe how user commands or environmental events change the state of the system.

In this paper, we describe our work on extending LTSs with state information, thus enabling a direct mapping of ADEPT models into extended LTSs. We also discuss how we extend our framework for HMI analysis to deal with such models. We applied the extended framework to an ADEPT autopilot model; this has been a challenging case study since our techniques could not scale to the size of the full model. Due to lack of space, we only summarise some of the results obtained from our analysis.

Related Work

Campos et al. (2011; 2008) propose a framework for analysing HMI using model-checking. They define a set of generic usability properties (Campos and Harrison 2008), such as the possibility to undo an action. These properties can be expressed in a modal logic called MAL and checked with a model checker on the system. This approach targets specific, and precise usability properties whereas our approach uses a more generic definition of good systems, and is complementary to their analysis.

Thimbleby et al. (2007; 2010) use graphs to represent models. They study usability properties of the system by analysing structural properties of graphs like the maximum degree and the value of centrality measures. In their ap-

proach, there is no distinction among actions and there is little focus on the dynamic aspects of the interaction.

Curzon et al. (2007) use a framework based on defining systems with modal logic. Properties of the model are checked using a theorem prover. Similarly to Campos et al., properties of interest are more targeted to a specific usability property while our approach is more generic.

Navarre et al. (2001; 2003) also developed a framework to analyse interactive systems. Their focus is on the combination of user task models and system models. We focus mainly on the system model although we have also explored an approach for checking whether a user task is supported by a system model (Combéfis 2009).

Bolton et al. (2008; 2011; 2010) developed a framework used to help predicting human errors and system failures. Models of the system are analysed against erroneous human behaviour models. The analysis is based on task-analytic models and taxonomies of erroneous human behaviour. All those models are merged into one model which is then analysed by a model checker to prove that some safety properties are satisfied.

Bredereke et al. (2002; 2005) formalised mode confusions and developed a framework to reduce them. The formalisation is based on a specification/implementation refinement relation. Their work is targeted on mode confusion while the work presented here is targeted to more general controllability issues.

Model-based testing has been used to analyse systems modelled as Input-Output Labelled Transition Systems (IOTS) (Tretmans 2008). The IO conformance relation (IOCO) is defined to describe the relationship between implementations and specifications. The IOCO relation states that the outputs produced by an implementation must, at any point, be a subset of the corresponding outputs in the specification. This is triggered by the fact that IOCO is used in the context of testing implementations. Outputs are similar to observations in our context. The full-control property defined in our work needs to consider commands (inputs) in addition to observations.

Modelling and Analysing HMI systems

This section provides a brief overview of our previously developed analysis framework for labelled transition systems (Combéfis et al. 2011a). In our framework, system and mental models are modelled with enriched LTSs called HMI-LTSs, which are essentially graphs whose edges are labelled with actions. The difference with classical LTSs is that three kinds of actions are defined; such distinction matters when concerned with controllability properties of systems (Heymann and Degani 2007; Javaux 2002):

1. *Commands* are actions triggered by the user on the system; they are also referred to as inputs to the system;
2. *Observations* are actions autonomously triggered by the system but that the user can observe; they are also referred to as outputs from the system;
3. *Internal actions* are neither controlled nor observed by the user; they correspond to internal behaviour of the system that is completely hidden to the user.

Formally, HMI-LTSs are tuples $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow \rangle$ where S is the set of states, $\mathcal{L}^c$ and $\mathcal{L}^o$ are the sets of commands and observations respectively, s_0 is the initial state and $\rightarrow \subseteq S \times (\mathcal{L}^c \cup \mathcal{L}^o \cup \{\tau\}) \times S$ is the transition relation. Internal actions cannot be distinguished by the user and are thus denoted with the same symbol τ, called the internal action. The set of observable actions comprises commands and observations and is denoted $\mathcal{L} = \mathcal{L}^c \cup \mathcal{L}^o$.

When a transition exists between states s and s' with action α, i.e., $(s, \alpha, s') \in \rightarrow$, we say that the action α is *enabled* in state s and we write $s \xrightarrow{\alpha} s'$. An HMI-LTS is *deterministic* if it has no internal transitions and its transition relation is a function from $(S \times (\mathcal{L}^c \cup \mathcal{L}^o))$ to S. In other words, if $s \xrightarrow{\alpha} s_1$ and $s \xrightarrow{\alpha} s_2$, then $s_1 = s_2$. The set of commands that are *enabled* in a state s, denoted $\Gamma^c(s)$, contains all the commands α such that there exists a state s' with $s \xrightarrow{\alpha} s'$. The set of enabled observations $\Gamma^o(s)$ is defined similarly.

Internal actions can occur between observable actions. A *weak transition* $s \stackrel{\alpha}{\Longrightarrow} s'$ corresponds to $s \xrightarrow{\tau^*\alpha\tau^*} s'$, where τ^* means zero, one or more occurrences of τ. The set of commands that are possible in a state s, denoted $A^c(s)$, corresponds to commands α such that there exists a state s' with $s \stackrel{\alpha}{\Longrightarrow} s'$. The set of possible observations, denoted $A^o(s)$, is defined similarly. A *trace* $\sigma = \langle \alpha_1, \ldots, \alpha_n \rangle$ is a sequence of observable actions in $\mathcal{L}$ that can be executed on the system, that is, such that $s_0 \stackrel{\alpha_1}{\Longrightarrow} s_1 \ldots s_{n-1} \stackrel{\alpha_n}{\Longrightarrow} s_n$. The set of traces of an LTS $\mathcal{M}$ is denoted $\mathbf{Tr}(\mathcal{M})$.

Full-control property

In our work, mental models are represented in terms of *deterministic* HMI-LTSs. They capture a simplified model of the system, as perceived by a human operator. We then define the *full-control property* to capture the fact that an operator's mental model carries enough information to enable proper control of the system. Full-control requires that, at any time during the interaction between the user and the system:

- the user knows exactly what are the possible commands on the system: the set of possible commands is the same on both the system and mental models;
- and the user is aware of at least the observations that can occur: the set of possible observations according to the mental model contains the ones possible on the system model.

Formally, a mental model $\mathcal{H} = \langle S_H, \mathcal{L}^c, \mathcal{L}^o, s_{0_H}, \rightarrow_H \rangle$ allows full-control of a given system $\mathcal{S} = \langle S_S, \mathcal{L}^c, \mathcal{L}^o, s_{0_S}, \rightarrow_S \rangle$ if and only if:

$$\forall \sigma \in \mathcal{L}^* \text{ such that } s_{0_S} \stackrel{\sigma}{\Longrightarrow} s_S \text{ and } s_{0_H} \xrightarrow{\sigma} s_H :$$
$$A^c(s_S) = A^c(s_H) \text{ and } A^o(s_S) \subseteq A^o(s_H) \qquad (1)$$

The existence of a full-control mental model for a given system model is guaranteed if the system is full-control deterministic, or *fc-deterministic*. This property, defined in (Combéfis and Pecheur 2009), requires that the non-determinism in a system does not interfere with its controllability; in other words, a system can be non-deterministic,

so long as it is still possible to construct a full-control mental model for it.

Interaction Analysis

Based on the full-control property, we proposed a framework and methodology to analyse systems from an HMI standpoint (Combéfis et al. 2011a). Two algorithms were developed in (Combéfis and Pecheur 2009; Combéfis et al. 2011b), which are focused on the automatic generation of a minimal full-control mental model for a given system. The first is based on the definition of a bisimulation-based relation between the states of the system, stating which of them can be merged together because they can be handled similarly from the standpoint of the operator. The second uses a learning algorithm which iteratively builds mental model guesses. The algorithm relies on a teacher to answer whether proposed execution sequences must, may or cannot be part of the mental model. The teacher uses the system model to answer such queries.

As the main input is a model of the system, the analysis can be performed and used in different steps of the HMI design process. In the evaluation phase, the analysis gives feedback regarding whether the model of the system is controllable by an operator. If it is not controllable, the analysis provides an example of a problematic interaction. The analysis can also be used at the end of the design process, once the system has been validated, in order to build artefacts such as user manuals, trainings, etc.

ADEPT

ADEPT (*Automatic Design and Evaluation Prototyping Toolset*) (Feary 2010) is a Java-based tool developed at NASA Ames, which supports designers in the early prototyping phases of the design of automation interfaces. The tool also offers a set of basic analyses that can be performed on the model under development. An ADEPT model is composed of two elements: a set of logic tables, coupled with an interactive user interface (UI). The logic tables describe the dynamics of the system as state changes in reaction to user actions or to environmental events. For example, Figure 1 shows a screenshot of the autopilot model opened in ADEPT. The left part of the window shows one of the logic tables and the right part shows the user interface.

The UI is composed of a set of components that are encoded as Java objects representing graphical widgets. The logic tables can refer to the elements of the UI and to the other components through their Java instance variables, and interact with them through their methods, using Java syntax. In particular, UI events are seen as boolean variables that are set to true when the event occurs.

Behind the scene, an ADEPT model is compiled into a Java program that can be executed in order to directly try the encoded behaviour with the user interface. That tool is meant to be used as a rapid prototyping tool. The models can then be tested and simulated by the designers, but can also be analysed by systematic and rigorous techniques. Possible analyses include validity checks on the structure of logic tables, for example. Our aim is to extend the analysis capabilities of ADEPT with our framework, which requires the translation of ADEPT tables into the models that our framework can handle.

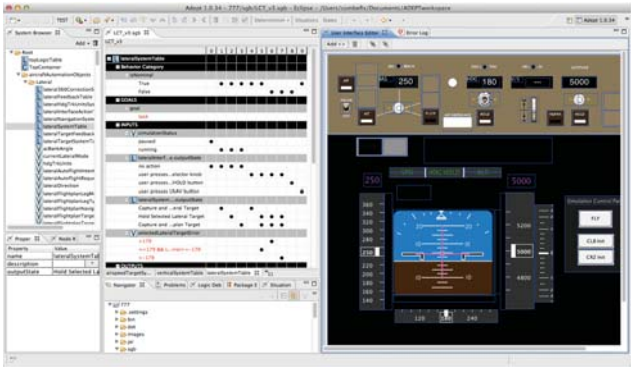

Figure 1: The autopilot model opened in ADEPT, with one logic table in the left part of the window and the user interface on the right part.

Figure 2 shows one of the logic tables of the autopilot model. The table example illustrates the way it can interact with elements of the UI. Each light grey line of the table corresponds to a variable of the system. The variables can be related to a component of the UI (such as pfdAirspeedTargetTape.currentValue), or they can be state variables of the model (such as indicatedAirSpeed) or they can relate to the internal logic of the system (airspeedSystemTable.outputState). The latter kind of variables can be seen as a description of the mode of a particular component of the system (the airspeed part in this example). For example, the two first lines of the output part of the logic table example mean that the value of the currentValue field of the pfdAirspeedTargetTape component of the UI is updated with the value of the indicatedAirspeed state variable. Moreover, each column of an ADEPT table corresponds to a transition scenario. From any state of the system that satisfies the condition described by the input part of the table, the system can move to the state of the system that results in applying the update instructions described by the output part of the table.

State Event Models

As already mentioned, HMI-LTSs are event-based, meaning that their states contain no information except for the behaviour in terms of events enabled in these states. On the other hand, as discussed above, ADEPT models combine state with transition information. Systems are composed of a set of variables, each system state corresponding to an assignment of values to the variables. Some variables are observable through the user interface, and tables describe the possible transitions from each state as a result of reacting to user commands or environmental events. A direct translation of ADEPT models into HMI-LTSs proved challenging, and for this reason we decided to extend HMI-LTSs by adding information on the states in the style of Kripke structures (Clarke Jr., Grumberg, and Peled 1999). In order to represent observable information on states, HVSs enrich HMI-LTSs with a set of *state-values* and with a mapping

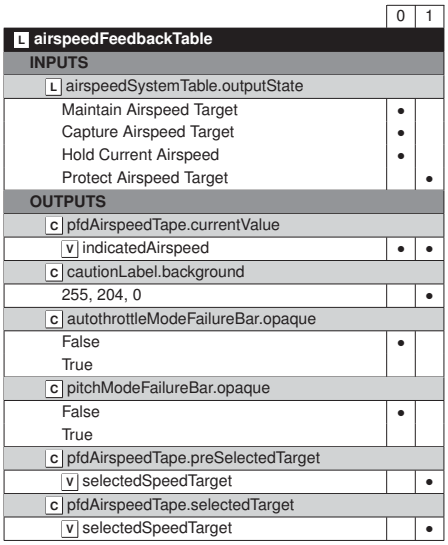

Figure 2: An example of a logic table: the airspeed feedback table of the autopilot model contains the logic related to the update of the UI for the airspeed part.

function associating each state to one state-value.

Definition 1 (HMI state-Valued System model (HVS)) *A* HMI state-valued system model *(HVS) is a tuple* $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow, \mathcal{L}^v, \mathcal{O}\rangle$ *where* $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow\rangle$ *is an HMI-LTS,* $\mathcal{L}^v$ *is a finite set of state-values and* $\mathcal{O} : S \mapsto \mathcal{L}^v$ *is a state-value mapping function. The three sets* $\mathcal{L}^c$*,* $\mathcal{L}^o$ *and* $\mathcal{L}^v$ *are disjoint.*

The set of state-values $\mathcal{L}^v$ and function $\mathcal{O}$ describe, for each state, the observations that can be made by the operator when the system is in that state. The difference between the state and transition related observations is in the interpretation according to a human-machine interaction point of view. The observations that can be made on the state may be ignored by the operator while interacting with the system. In contrary, observations occurring on a transition are output by the system and are expected to be seen by the operator for the interaction to proceed according to the chosen definition. Note that we do not model the possibility for the operator to get distracted and miss the observation.

Mental models are similarly enriched to include state information. Observations can be event-based (observations on transitions) or state-based (state-values on states). State-values are taken into account in the human model by *action guards*, that is, conditions on the state-value that must be verified in the current state of the system. In order to represent mental models, HVMs are HMI-LTSs enriched with state-values on the transitions. Moreover, as already stated, mental models are considered deterministic and free of τ-transitions in this work.

Definition 2 (HMI state-Valued Mental model (HVM)) *A* HMI state-valued mental model *(HVM) is a tuple* $\langle S, \mathcal{L}^c, \mathcal{L}^o, s_0, \rightarrow, \mathcal{L}^v\rangle$ *where* $\mathcal{L}^v$ *is a finite set of state-values,* $\rightarrow \subseteq S \times \mathcal{L}^v \times \mathcal{L} \times S$ *and* $\langle S, \mathcal{L}^v \times \mathcal{L}^c, \mathcal{L}^v \times \mathcal{L}^o, s_0, \rightarrow\rangle$ *is a deterministic HMI-LTS without* τ*-transition. The three sets* $\mathcal{L}^c$*,* $\mathcal{L}^o$ *and* $\mathcal{L}^v$ *are disjoint.*

HVSs and HVMs do not handle state observation in the same way. That distinction is a result of the meaning of both models in terms of human-machine interaction. The operator must always have the ability to make an observation on the machine, depending on its state. Concerning the mental model, the state-value is used to condition the actions that the operator may execute. In one given state, the operator may have several actions with guarded with different state-values.

For an HVS and HVM that share the same alphabet of actions and state-observations, their interaction is defined as follows:

Definition 3 (Interaction between an HVS and HVM) *Given an HVS* $\mathcal{S} = \langle S_S, \mathcal{L}^c, \mathcal{L}^o, s_{0_S}, \rightarrow_S, \mathcal{L}^v, \mathcal{O}\rangle$ *and an HVM* $\mathcal{H} = \langle S_H, \mathcal{L}^c, \mathcal{L}^o, s_{0_H}, \rightarrow_H, \mathcal{L}^v\rangle$*, the interaction between* $\mathcal{S}$ *and* $\mathcal{H}$*, denoted* $\mathcal{S} \parallel_I \mathcal{H}$ *, is an LTS* $\mathcal{I} = \langle S_I, \mathcal{L}^c \cup \mathcal{L}^o, s_{0_I}, \rightarrow_I\rangle$ *where* $S_I \subseteq (S_S \times S_H)$*,* $s_{0_I} = (s_{0_S}, s_{0_H})$ *and* $\rightarrow_I \subseteq S_I \times (\mathcal{L}^c \cup \mathcal{L}^o \cup \{\tau\}) \times S_I$ *is defined so that:*

- $(s_{S_1}, s_{H_1}) \xrightarrow{\alpha} (s_{S_2}, s_{H_2})$ *if and only if* $s_{S_1} \xrightarrow{\alpha} s_{S_2}$ *and* $s_{H_1} \xrightarrow{[v]\,\alpha} s_{H_2}$ *with* $v = \mathcal{O}(s_{S_1})$
- *and* $(s_{S_1}, s_{H_1}) \xrightarrow{\tau} (s_{S_2}, s_{H_1})$ *if and only if* $s_{S_1} \xrightarrow{\tau} s_{S_2}$.

Intuitively, the operator can perform a visible action (command or observation) if the state-value of the current state of the system agrees with the action guard that is present on the mental model.

For enriched models, the full-control property is defined based on the above interaction model. The difference with the definition based on HMI-LTS is that possible commands and observations have to be considered in pair with the associated state-value (from the source state for HVSs and on the action guard for HVMs).

Definition 4 (Full-Control Property) *For HVS* $\mathcal{S} = \langle S_S, \mathcal{L}^c, \mathcal{L}^o, s_{0_S}, \rightarrow_S, \mathcal{L}^v, \mathcal{O}\rangle$ *and HVM* $\mathcal{H} = \langle S_H, \mathcal{L}^c, \mathcal{L}^o, s_{0_H}, \rightarrow_H, \mathcal{L}^v\rangle$*,* $\mathcal{H}$ *is said to* allow full-control *of* $\mathcal{S}$*, which is denoted* $\mathcal{H}$ **fc** $\mathcal{S}$*, if and only if for all reachable* $(s_S, s_H) \in \mathcal{S} \parallel_I \mathcal{H}$*:*

- $A^c(s_S) = A^c(s_H)$*;*
- *and* $A^o(s_S) \subseteq A^o(s_H)$

where $A^c(s) = \{(v, c) \mid \exists s \xrightarrow{\tau^*} s' \xrightarrow{c} s'' \wedge v = \mathcal{O}(s') \wedge c \in \mathcal{L}^c\}$ *for HVSs and* $A^c(s) = \{(v, c) \mid \exists s \xrightarrow{[g]c} s' \wedge v \models g \wedge c \in \mathcal{L}^c\}$ *for HVMs. The* $A^o(s)$ *set is defined similarly.*

Enriched models can be expanded into HMI-LTSs so that the full-control property is preserved. The motivation for such an expansion is to make it possible to perform the same reasoning and analyses that can be done on HMI-LTSs. The expansion operation transforms state-values into observation actions.

The expansion of an HVS is based on the mapping shown on Figure 3. Every non-τ transition $s \xrightarrow{\alpha} t$ is expanded so that the state-value v is checked before the occurrence of the action, that is, a new state s^v is added to the expanded model with the sequence of transitions $s \xrightarrow{v} s^v \xrightarrow{\alpha} t$. For

τ-transitions, no transformation occur, they are preserved in the expanded HMI-LTS.

Figure 3: Transition mapping for HVS to HMI-LTS translation for non-τ transitions. The transition from the original system model (on the left) induces two transitions in the expanded system model (on the right), where $v = \mathcal{O}(s)$.

Definition 5 (Expansion of an HVS to an HMI-LTS) *The expansion $exp(\mathcal{S})$ of HVS $\mathcal{S} = \langle S_S, \mathcal{L}^c, \mathcal{L}^o, s_{0_S}, \rightarrow_S, \mathcal{L}^v, \mathcal{O}\rangle$, is an HMI-LTS $\mathcal{E} = \langle S_E, \mathcal{L}^c, \mathcal{L}^o_E, s_{0_S}, \rightarrow_E\rangle$ where $\mathcal{L}^o_E = \mathcal{L}^o \cup \mathcal{L}^v$ and:*

- $S_E = S_S \cup \{s^v \mid s \in S_S, v = \mathcal{O}(s)$ *and* $\Gamma(s) \neq \emptyset\}$;
- $\rightarrow_E = \{(s, \tau, t) \mid (s, \tau, t) \in \rightarrow_S\}$ $\cup \{(s, v, s^v), (s^v, \alpha, t) \mid (s, \alpha, t) \in \rightarrow_S$ *and* $v = \mathcal{O}(s)\}$.

By construction, the states of the expanded HMI-LTS can be partitioned into two sets:

- *Observation states* are those from the original HVS and intuitively correspond to the fact that the operator must check the state-value before the occurrence of a visible action on the system. Those states can have outgoing τ-transitions and at most one outgoing transition labelled with a state-value. An observation state s is characterised by $|A^o(s)| = 1, A^o(s) \subseteq \mathcal{L}^v$ and $A^c(s) = \emptyset$.
- *Action states* are the states added to the HVS. Those states have outgoing transitions with commands and observations that correspond to those from the original HVS. They do not have any τ-transitions. An action state s is characterised by $A(s) \subseteq \mathcal{L}$.

Transitions outgoing from an observation state lead to an action state, except for τ-transitions that lead to another observation state. And conversely, transitions outgoing from an action state always lead to an observation state.

HVMs can also be expanded into HMI-LTSs, so as to make enriched traces explicit. The expansion is based on the intuition that the operator must always first check whether the action guard is satisfied before doing any visible action. Figure 4 shows the mapping between the transitions from the HVM and the corresponding expanded HMI-LTS. Every transition $s \xrightarrow{[v]\alpha} s'$ is expanded so that the state-value v is checked before performing the α action, that is, a new state s^v is added to the expanded model with the sequence of transitions $s \xrightarrow{v} s^v \xrightarrow{\alpha} s'$.

Figure 4: Transition mapping for HVM expansion. The transition from the HVM (on the left) induces two transitions in the expanded HMI-LTS (on the right).

Definition 6 (Expansion of an HVM to an HMI-LTS) *The expansion $exp(\mathcal{H})$ of an HVM $\mathcal{H} = \langle S_H, \mathcal{L}^c, \mathcal{L}^o, s_{0_H}, \rightarrow_H, \mathcal{L}^v\rangle$, is an HMI-LTS $\mathcal{E} = \langle S_E, \mathcal{L}^c, \mathcal{L}^o_E, s_{0_H}, \rightarrow_E\rangle$ where $\mathcal{L}^o_E = \mathcal{L}^o \cup \mathcal{L}^v$ and:*

- $S_E = S_H \cup \{s^v \mid s \in S_H, (s, v, \alpha, t) \in \rightarrow_H\}$;
- $\rightarrow_E = \{(s, v, s^v), (s^v, \alpha, t) \mid (s, v, \alpha, t) \in \rightarrow_H\}$.

Similarly to HVSs, two kinds of states are identifiable by construction: observation states and action states. The observation states intuitively correspond to the check of the action guard by the user. They are characterised by $\Gamma(s) \subseteq \mathcal{L}^v$ and their outgoing transitions always lead to an action state. The action states correspond to those from the HVM. They are characterised by $\Gamma(s) \subseteq \mathcal{L}$. Their outgoing transitions are labelled with commands and observations and always lead to an observation state.

According to the following theorem, proven in (Combéfis 2013), it is possible to reduce the check that an HVM allows full control of an HVS, to the equivalent check that the expanded mental model allows full-control of the expanded system model.

Theorem 7 *Given an HVS $\mathcal{S} = \langle S_S, \mathcal{L}^c, \mathcal{L}^o, s_{0_S}, \rightarrow_S, \mathcal{L}^v, \mathcal{O}\rangle$ and an HVM $\mathcal{H} = \langle S_H, \mathcal{L}^c, \mathcal{L}^o, s_{0_H}, \rightarrow_H, \mathcal{L}^v\rangle$:*

$$\mathcal{H}\ \mathbf{fc}\ \mathcal{S} \iff exp(\mathcal{H})\ \mathbf{fc}\ exp(\mathcal{S})$$

Experience with the Autopilot model

The ADEPT autopilot partially models the behaviour of the autopilot of a Boeing 777 aircraft. The full autopilot ADEPT model has a total of 38 logic tables. Three major groups of tables can be identified in the model, namely one for the lateral aspect, one for the vertical aspect and finally one for the airspeed aspect. For each of these aspects, the logic tables are further partitioned into three groups: the action tables, the system tables and the feedback tables, successively executed in that order. Action tables determine actions from UI events, system tables update the state of the system according to the performed action, and feedback tables reflect the state of the system to UI elements. Our analyses focus on the system tables.

Due to space limitations, we are not able to fully report on our experience with the autopilot case study. Details of the system and our analysis results can be found in (Combéfis 2013). However, we outline in this section some of the results we obtained and observations we made.

First of all, by extending our HMI models with state information, we were able to automatically translate the autopilot tables, which would have been very hard to perform reliably by hand, given the size of the system. Moreover, our techniques cannot scale to the full size of such a large and complex model, the main reason being related to the large number of system variables with large domains causing a state explosion issue. Therefore, our analyses were performed on parts of the autopilot model, each analysis considering subsets of the system tables. As is typical with formal techniques, we additionally had to abstract the infinite data domains involved in the models.

Using the outputState as a mode indicator, we performed mode confusion analysis, and detected a potential mode confusion on the airspeedSystemTable.

This was identified during the minimal model generation phase, where the generation algorithm produced an error trace witnessing the fact that the system model was not fc-deterministic. By analysing the error trace manually, we localised the erroneous behaviour in the involved ADEPT tables.

One of the models that was analysed was an HVS with 7680 states and 66242 transitions, among which 57545 are labelled with commands and 8697 are internal τ-transitions. The obtained minimal mental model has 25 states and 180 transitions.

Finally, the semantics and translation algorithms that we have developed are only a first step towards effectively integrating our techniques with ADEPT. While we have worked on translating ADEPT models to HVSs, we still need to be able to directly relate the results of our analyses with the original ADEPT models. In the future, we therefore plan on working on a better integration of our analysis tools with ADEPT, better visualisation of the analysis results, and on increasing scalability of our analysis algorithms.

References

Bastide, R.; Navarre, D.; and Palanque, P. 2003. A tool-supported design framework for safety critical interactive systems. *Interacting with Computers* 15(3):309–328.

Bolton, M., and Bass, E. 2010. Using task analytic models and phenotypes of erroneous human behavior to discover system failures using model checking. In *Proceedings of the 54th Annual Meeting of the Human Factors and Ergonomics Society*, 992–996.

Bolton, M.; Bass, E.; and Siminiceanu, R. 2008. Using formal methods to predict human error and system failures. In *Proceedings of the Second International Conference on Applied Human Factors and Ergonomics (AHFE 2008)*.

Bolton, M.; Siminiceanu, R.; and Bass, E. 2011. A systematic approach to model checking human-automation interaction using task analytic models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* 41(5):961–976.

Bredereke, J., and Lankenau, A. 2002. A rigorous view of mode confusion. In *Proceedings of the 21st International Conference on Computer Safety, Reliability and Security (SAFECOMP 2002)*, 19–31. Springer.

Bredereke, J., and Lankenau, A. 2005. Safety-relevant mode confusions — modelling and reducing them. *Reliability Engineering & System Safety* 88(3):229–245.

Campos, J. C., and Harrison, M. D. 2008. Systematic analysis of control panel interfaces using formal tools. In Graham, T. N., and Palanque, P. A., eds., *Proceedings of the 15th International Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS 2008)*, volume 5136 of *Lecture Notes in Computer Science*, 72–85. Springer.

Campos, J. C., and Harrison, M. D. 2011. Model checking interactor specifications. *Automated Software Engineering* 8(3):275–310.

Clarke Jr., E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. The MIT Press.

Combéfis, S., and Pecheur, C. 2009. A bisimulation-based approach to the analysis of human-computer interaction. In Calvary, G.; Graham, T. N.; and Gray, P., eds., *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2009)*, 101–110. New York, NY, USA: ACM.

Combéfis, S.; Giannakopoulou, D.; Pecheur, C.; and Feary, M. 2011a. A formal framework for design and analysis of human-machine interaction. In *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2011)*, 1801–1808. IEEE.

Combéfis, S.; Giannakopoulou, D.; Pecheur, C.; and Feary, M. 2011b. Learning system abstractions for human operators. In *Proceedings of the 2011 International Workshop on Machine Learning Technologies in Software Engineering (MALETS 2011)*, 3–10. New York, NY, USA: ACM.

Combéfis, S. 2009. Operational model: Integrating user tasks and environment information with system model. In *Proceedings of the 3rd International Workshop on Formal Methods for Interactive Systems*, 83–86.

Combéfis, S. 2013. *A Formal Framework for the Analysis of Human-Machine Interactions*. Ph.D. Dissertation, Université catholique de Louvain.

Curzon, P.; Sėnas, R. R.; and Blandford, A. 2007. An approach to formal verification of human-computer interaction. *Formal Aspects of Computing* 19(4):513–550.

Feary, M. S. 2010. A toolset for supporting iterative human–automation interaction in design. Technical Report 20100012861, NASA Ames Research Center.

Heymann, M., and Degani, A. 2007. Formal analysis and automatic generation of user interfaces: Approach, methodology, and an algorithm. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 49(2):311–330.

Javaux, D. 2002. A method for predicting errors when interacting with finite state systems. How implicit learning shapes the user's knowledge of a system. *Reliability Engineering and System Safety* 75:147–165.

Navarre, D.; Palanque, P.; and Bastide, R. 2001. Engineering interactive systems through formal methods for both tasks and system models. In *Proceedings of the RTO Human Factors and Medicine Panel (HFM) Specialists' Meeting*, 20.1–20.17.

Thimbleby, H., and Gow, J. 2007. Applying graph theory to interaction design. In Gulliksen, J.; Harning, M. B.; Palanque, P.; van der Veer, G.; and Wesson, J., eds., *Proceedings of the Engineering Interactive Systems Joint Working Conferences EHCI, DSV-IS, HCSE (EIS 2007)*, volume 4940 of *Lecture Notes in Computer Science*, 501–519. Springer.

Thimbleby, H. 2010. *Press On: Principles of Interaction Programming*. The MIT Press.

Tretmans, J. 2008. Model based testing with labelled transition systems. In Hierons, R.; Bowen, J.; and Harman, M., eds., *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*. Springer. 1–38.