# Integrating Automated Verification into Interactive Systems Development*

José C. Campos†
HCI Group, Department of Computer Science, University of York, York, UK
Jose.Campos@cs.york.ac.uk

## Abstract

*Our field of research is the application of automated reasoning techniques during interactor based interactive systems development. The aim being to ensure that the developed systems embody appropriate properties and principles. In this report we identify some of the pitfalls of current approaches and propose a new way to integrate verification into interactive systems development.*

## 1. Introduction

The widespread use of computers puts increasing demands on user interfaces. On the one hand, systems must be intuitive and easy to use, on the other hand they must ensure safety and avoid risk. Due to their increasing complexity, reasoning about systems behaviour has become increasingly hard. This raises the question of how to ensure quality during development.

The use of formal methods has long since been proposed as a solution to this problem. The advantages are two-fold: they enable better design understanding and communication; and mathematical reasoning can be used to validate the design. This last point is especially useful when we think of ensuring system quality, as it allows us to assess the system from early stages in the development process.

Because reasoning about specifications of complex systems will be a complex and error prone exercise in itself, ways of automating the reasoning process have been sought. Two well established approaches to automated reasoning are model checking [7] and theorem proving. While these techniques have been used mainly in the field of hardware verification [8], their application to the verification of reactive systems in general is also being studied [18].

Despite being a particular case of reactive systems, interactive systems have specific concepts and concerns. So, novel approaches have been sought. In this context, the notion of Interactor [12, 20] has been introduced as a way to structure specifications of interactive systems.

Our field of research is the application of automated reasoning techniques during interactor based interactive systems development. The aim being to ensure that the developed systems embody appropriate properties and principles.

## 2. Review

Four major approaches to the formal (automated) verification of interactive systems have been identified so far [4, 6]. Three use model checking: Abowd, Wang & Monk [1] use SMV [19], Paternó [20] uses the Lite tool-set [17], d'Ausbourg, Durrieu & Roche [9] use a model checking related technique based on Lustre; and one uses theorem proving: Bumbulis [3] uses the HOL theorem prover.

In order to better compare these approaches we have defined a framework with which to compare them [6]. It identifies three entities involved in interaction: *User*, *User Interface*, and an *Underlying System*. Interaction proceeds through interaction mechanisms: *Events* and *Status Phenomena* are atomic, *Task*, and *Mode* are used to structure the user interface. The framework identifies also three basic types of properties to be verified: *Visibility*, *Reachability*, and *Reliability*. Table 1 summarises the results of the review in terms of what each approach addresses (✓), partially addresses (∼), or does not address (×).

The conclusions drawn from the review are two-fold. At the technological level, it was seen that both model checking and theorem proving have difficulties when dealing with the added complexity introduced by interactive systems.

At the methodological level, there is a need to further investigate what should/can be proved of interactive systems using automated reasoning tools. Previous approaches have tried to map what could be expressed in traditional verification tools into the *interactive systems space*. In order to make the most of automated reasoning we must try to do the opposite: identify what properties are interesting and map

*Published In *13th IEEE International Conference: Automated Software Engineering — Doctoral Symposium Proceedings*, pages 13–15. October 1998.

†José Campos is supported by Fundação para a Ciência e a Tecnologia (FCT, Portugal) under grant PRAXIS XXI/BD/9562/96.

**Table 1. Summary of the comparison**

| | | SMV | Lotos | Lustre | HOL |
|---|---|---|---|---|---|
| Entities | Users | × | × | × | × |
| | User Interf. | ∼ | ✓ | ✓ | ✓ |
| | Underl. Sys. | ∼ | × | × | × |
| Inter. Mech. | Events | ∼ | ✓ | ✓ | ∼ |
| | Stat. phenom. | ∼ | × | ✓ | ∼ |
| | Modality | × | ✓ | × | × |
| | Task | ∼ | ∼ | × | × |
| | Mode | × | × | × | × |
| Prop. | Visibility | × | ✓ | ✓ | × |
| | Reachability | ✓ | ✓ | ✓ | × |
| | Reliability | ✓ | ∼ | ∼ | ∼ |

them into automated verification tools.

If we combine the above two concerns, we can identify a third issue that needs addressing: when should we do the proofs? — i.e., at what level of abstraction, and at what stage of development should we be working? Traditionally, verification has been used to assess design against absolute measures of quality. Regarding HCI, matters are not so clear cut. Furthermore, if we are using principled design, it would be useful to test the design decisions against the appropriate principles as soon as possible.

The challenge, then, is trying to make the best of the available verification technology by means of defining an appropriate methodological framework which will allow us to identify how and when verification should be applied.

## 3. The Thesis

In view of the complexity of the systems, and of the limitations of the available technology, the best approach to achieve the goals set forth above is to allow for a flexible scheme of verification. With this in mind we established the following objectives:

- non commitment to a specific technique — we want to be able to use model checking and theorem proving as appropriate, and not to be tied to a particular verification strategy.
- use of partial models — models that try to address all relevant aspects of an interactive system are too complex; instead, we want to use partial models, each model focusing on different design aspects (cf. [13]).

These two points, together with the observation that identifying (let alone proving) interesting properties in "finished" models becomes difficult, lead us to the realization that instead of being used as a *post facto* check on the quality of the specification, verification should be used to inform design decisions during development [5]. This can be done by using partial models that highlight the design features under consideration, and allows us to use the most appropriate verification technique for each model.

All the above leads to the definition of four lines of work:

- verification as a support to design — verification should be used to inform design decisions rather than to check the final design;
- understanding properties — we need to establish a framework that enables us to reason about how to go from design principles to verifiable properties;
- model checking for interactor specifications — we need to determine how model checking can be applied to interactor based specifications;
- theorem proving for interactor specifications — similarly, we need to determine how theorem proving can be applied to interactor based specifications;

and the definition of the central proposition of the thesis as: *Formal verification techniques (automated reasoning tools in particular) can be used to inform design decisions during interactive systems development.*

A novel approach to the integration of automated verification into interactive systems development will be proposed, and it will be shown how model checking and theorem proving can be used in the context of the approach.

## 4. Progress

In this section, we briefly describe the work done so far.

### 4.1. The role of formal verification

We propose that verification should be used to inform design choices during development, and not only as a check on the correctness of the specified system. The complete rationale behind this proposal is presented in [5]. Some of the points that are made are: that the role properties play depends not only on the system under consideration, but also on the particular specification that is adopted; that it is difficult to base design decisions on prescriptive theories alone, so the possibility of early assessment of design decisions would be useful; that seeing the verification step as a final step in the development process, and trying to use *off the shelf* properties, might lead us to end up looking at properties of the specification instead of the system; and finally, that the particular specification style adopted influences which verification tools can be used.

The use of verification to inform design can be achieved by using, not a monolithic specification which tries to encompass all of the system, but a set of models each focusing on particular features of the system. This type of approach

has a number of benefits. Namely: we use verification to validate the choices that are made in relation to what is important of the system, not its specification; we are able to apply the most appropriate verification technique in each case; conversely, we can develop each model in the most suitable way, regarding the tool that will be used; also, using models that focus on properties means we will be able to verify properties that otherwise would be too difficult to check; finally, we might be able to reuse the proofs when thinking of related properties of different systems.

## 4.2. Using model checking

We are exploring the use of model checking in the verification of interactor based specifications. This is being done at two levels: using a traditional model checker (SMV [19]), and using the $\mu$-calculus model checker in PVS.

### 4.2.1 SMV

A compiler has been developed (see [5]) that enables us to analyse Interactors specifications in SMV. For an introduction to Interactors see [12]. In short, interactors are objects which allow their state to be perceived through some presentation (cf. **visible** clause below). Interactors provide a framework for specification and do not prescribe a particular notation.

In the present case, we are using Modal Action Logic (MAL) [21] to specify interactors behaviour. In the input language accepted by the compiler, an interactor describing whether a window is mapped on the screen looks like this:

**interactor** $window$
**attributes**
$mapped : boolean$
**visible**
$mapped$
**actions**
$map, unmap$
**axioms**
1. $[]\neg\ mapped$
2. $\neg\ mapped \Rightarrow [map]\mathsf{next}(mapped)$
3. $mapped \Rightarrow [unmap]\neg\ \mathsf{next}(mapped)$

Besides the clauses shown in the example, the interactor notation allows for three additional clauses: **importing** (allows inheritance), **fairness** (allows the definition of a fairness expression to be used by SMV), and **define** (enable us to give names to expressions as can be done in SMV). Multiple interactor specifications can be written by organising interactors in a hierarchy. In order to translate these hierarchies of interactors into SMV, we use the notion of module. So, each interactor will be a module in SMV.

To test properties of the specification, a further clause was introduced in the language: **test**. It is used to specify a CTL formula whose validity is to be verified by SMV.

In [5] it is shown how the compiler and SMV can be used to reason about different design possibilities in the development of an e-mail client.

### 4.2.2 PVS

PVS comes with a theory that defines the CTL operators in terms of the $\mu$-calculus. Alternatively we can define temporal operators for other logics. In [6, Appendix C] we have defined the operators for ACTL.

In order to use the model checker, the specification needs to be structured as a predicate over pairs of states, where the state type must be finite. We can then use the temporal operators to write putative theorems. PVS performs BDD simplification over the finite-state machine defined by the predicate over pairs of states, rewrites the temporal operators in terms of $\mu$-calculus, and runs the resulting state machine and $\mu$-calculus predicate in the model checker.

The present approach to model interactors and properties in this way is still tentative. We plan to expand on it in order to explore how the combination of theorem proving and model checking can be used to enhance the analytic power of both techniques.

## 4.3. Using theorem proving

While theorem provers do not have facilities to perform temporal reasoning, they are better than model checkers when it comes to reasoning over more information oriented features of systems. At the moment, three possible uses for theorem proving are envisaged: the validation of the adequacy of perceptual operators as suggested in [11] (see below), using it as an additional layer over model checking, and embedding a temporal logic in PVS (c.f. [16]).

### 4.3.1 Perceptual operators

The type of analysis described in [11] has to do mainly with symbolic manipulation of expressions in order to prove their equality. The basic idea is that properties that are proved of an abstract specification must also be shown to held at the level of the concrete presentation of the system. To do this we represent both a model of the abstract specification of the system and a model of the concrete presentation that is proposed as PVS theories, and then use PVS to determine if predicates over the abstract model are equivalent to corresponding predicates over the presentation model.

In [10] we apply this line of reasoning to the analysis of an aircraft air speed indicator, regarding its fitness to assist the pilot in the task of maintaining the correct aircraft configuration during landing (cf. [15]). Three PVS theories were developed. One to specify the logical model of the air speed indicator as well as the logical operators that support the task; another to specify the concrete circular air

speed indicator, with its needle and speed bugs (which indicate at which air speeds the aircraft configuration should be changed), and the mapping from the logical to the perceptual level; and finally a third theory which introduces the conjectures to be verified. As an example we present here one of the conjectures which is analysed in [10]:

$$\begin{aligned}&\mathsf{configuration\_change\_task} : \mathsf{CONJECTURE}\\ &\mathsf{configChangeCheck}(\mathsf{abs\_asi}) =\\ &\mathsf{asiConfigCheck}(\rho(\mathsf{abs\_asi}))\end{aligned}$$

What this conjecture expresses is that checking for the need to change the aircraft configuration should yield the same result regardless of the check being done at the logical level (configChangeCheck) or at the perceptual level (asiConfigCheck). $\rho$ is the mapping from the logical to the perceptual level.

In [10] we show how performing this type of consistency check improves our understanding of the specification, and allows us to identify assumptions about the system which are embedded in the representation but not made explicitly represented anywhere. As an example, during the proof of the conjecture above, we were led to realize how, at the presentation level, the speed bugs implicitly acquire the function of indicating the aircraft current configuration.

## 5. Conclusion

We have motivated the field of formal (automated) verification of interactive systems, and identified the main approaches to the area (see [4] for a more detailed review).

We have identified some of the pitfalls of the current approaches and proposed a new way to integrate verification into interactive systems development.

We have also briefly described the work done so far (see also [6, 5, 10] for more details).

## References


[1] Gregory D. Abowd, Hung-Ming Wang, and Andrew F. Monk. A formal technique for automated dialogue development. In *Proceedings of the First Symposium of Designing Interactive Systems - DIS'95*, pages 219–226. ACM Press, August 1995.

[2] F. Bodart and J. Vanderdonckt, editors. *Design, Specification and Verification of Interactive Systems '96*, Springer Computer Science. Springer-Verlag/Vien, June 1996.

[3] Peter Bumbulis. *Combining Formal Techniques and Prototyping in User Interface Construction and Verification*. PhD thesis, University of Waterloo, 1996.

[4] José C. Campos and Michael D. Harrison. Formal verification of interactive systems: A review. In Harrison and Torres [14], pages 109–124.

[5] José C. Campos and Michael D. Harrison. The role of verification in interactive systems design. In *Design, Specification and Verification of Interactive Systems '98*, Springer Computer Science, pages 155–170. Eurographics, Springer-Verlag/Wien, 1998.

[6] José Creissac Campos. Formal verification of interactive systens. 1st year qualifying dissertation, Department of Computer Science, University of York, June 1997.

[7] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.

[8] Edmund M. Clarke and Jeannette M. Wing. Formal methods: state of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, December 1996.

[9] Bruno d'Ausbourg, Guy Durrieu, and Pierre Roche. Deriving a formal model of an interactive system from its UIL description in order to verify and to test its behaviour. In Bodart and Vanderdonckt [2], pages 105–122.

[10] G. Doherty, J. C. Campos, and M. D. Harrison. Representational reasoning and verification. In *Proceedings of the BCS-FACS Workshop: Formal Aspects of the Human Computer Interaction*, pages 193–212. Computing Research Centre, Sheffield Hallam University, September 1998.

[11] Gavin Doherty and Michael D. Harrison. A representational approach to the specification of presentations. In Harrison and Torres [14], pages 273–290.

[12] David J. Duke and Michael D. Harrison. Abstract interaction objects. *Computer Graphics Forum*, 12(3):25–36, 1993.

[13] Bob Fields, Nick Merriam, and Andy Dearden. DMVIS: Design, modelling and validation of interactive systems. In Harrison and Torres [14], pages 29–44.

[14] M. D. Harrison and J. C. Torres, editors. *Design, Specification and Verification of Interactive Systems '97*, Springer Computer Science. Springer-Verlag/Vien, June 1997.

[15] E. Hutchins. How a cockpit remembers its speed. *Cognitive Science*, 19:265–288, 1995.

[16] Pertti Kellomäki. *Mechanical Verification of Invariant Properties of DisCo Specifications*. PhD thesis, Tampere University of Technology, 1997.

[17] José A. Mañas et al. *Lite User Manual*. LOTOSPHERE consortium, March 1992. Ref. Lo/WP2/N0034/V08.

[18] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, 1995.

[19] K. L. McMillan. *The SMV system*. Carnegie-Mellon University, draft edition, February 1992.

[20] Fabio Paternó. *A Method for Formal Specification and Verification of Interactive Systems*. PhD thesis, Department of Computer Science, University of York, 1995.

[21] Mark Ryan, José Fiadeiro, and Tom Maibaum. Sharing actions and attributes in modal action logic. In T. Ito and A. R. Meyer, editors, *Theoretical Aspects of Computer Software*, volume 526 of *Lecture Notes in Computer Science*, pages 569–593. Springer-Verlag, 1991.