A Survey of Papers from Formal Methods for Interactive Systems (FMIS) Workshops

Pascal Béger, Sébastien Leriche, Daniel Prun

A Survey of Papers from Formal Methods for Interactive Systems (FMIS) Workshops

Pascal Béger , Sébastien Leriche , and Daniel Prun

ENAC, Université de Toulouse, France
{pascal.beger, sebastien.leriche, daniel.prun}@enac.fr
http://lii.enac.fr/

Abstract

Our research team is specialized in human-computer systems and their engineering, with focus on interactive software systems for aeronautics (from cockpits to control towers). This context stands out by the need for certification, such as DO-178 or ED-12. Today, formal methods are pushed forward, as one of the best tools to achieve the verification and validation of properties, leading to the certification of these systems.
Interactive systems are reactive computer systems that process information from their environment and produce a representation of their internal state. They offer new rich interfaces with sophisticated interactions. Their certification is a challenge, because the validation is often a human based process since traditional formal tools are not always suitable to the verification of graphical properties in particular.
In this paper, we explore the scientific work that has been done in formal methods for interactive systems over the last decade, in a systematic study of publications in the International Workshop on Formal Methods for Interactive Systems. We describe an analytical framework that we apply to classify the studied work into classes of properties and used formalisms. We then discuss the emerging findings, mainly the lack of papers addressing the formal specification or validation of perceptibility properties. We conclude with an overview of our future work in this area.

Keywords: Interactive software - Formal methods - Verification - Graphical properties

1 Introduction

1.1 Aim and scope of the article

Interactive systems are reactive computer systems that process information (mouse clicks, data entries, etc.) from their environment (other systems or human users) and produce a representation (sound notification, visual display, etc.) of their internal state [13, 59]. They now have an increasingly important place among modern systems in various sectors such as aeronautics, space, medical or mobile applications. These systems offer new rich human machine interfaces with sophisticated interactions.

The preferred method for the verification and validation (V&V) of properties on interactive systems remains largely based on successive testing sessions of prototypes, performed through various experimentations involving representative end-users. For a long time, formal methods have not been very used to the verification of interactive properties. Indeed, historically, formal methods have been developed for distributed and embedded systems. The first properties studied for software and computer systems concerned safety (e.g. absence of unwanted events, boundedness) as well as program liveness (e.g. return to a given state, deadlock freedom) [63]. The main methods used to verify and validate properties of systems are model verification by model checking [25], mathematical proof [18], static analysis [44] and test processes driven by a formal model of the system under tests.

However, more and more work is being done on the development of formal methods to interactive systems. The objective is to study how these methods can be adapted to the modelling and the verification of properties involving human related characteristics. In particular, in the scope of critical domains such as aeronautics, recent updates of standards used for certification strongly recommend to use formal methods for the verification and validation of requirements of new software for aircraft cockpits ([72, 73]).

In this context, the objective of this survey is to study research activities that have been done in formal methods for the modeling, verification and validation of interactive systems, over the last decade. The aim is to draw a faithful picture of formalisms that are used to model interactive systems, set of properties that are verified and formal methods applied. From this picture, the objective is to identify strengths and weaknesses of formal approaches for interactive systems and to identify ways of improvements. More precisely, the survey highlights several points: What interactive related properties are studied? Which ones are more covered and which ones are least addressed? Are there formalisms that are widely used to model systems and study their properties? Are there any new formalisms that have emerged? Are they used on industrial critical systems or only on small academic case studies?

1.2 Methodology

Through this survey we explore the scientific work that has been done in formal methods for interactive systems over the last decade. For this purpose, we perform a systematic study of publications from a specific workshop, the International Workshop on Formal Methods for Interactive Systems (FMIS). We have selected this workshop because it covers exactly our problematic: the articles from this workshop address issues of how formal methods can be applied to interactive system design and verification and validation of their related properties. The workshop also focuses on general design and verification methodologies, and takes models and human behavior under consideration. Moreover, FMIS has reached a critical mass that makes the analysis more significative and reliable. It has taken place seven times from 2006 to 2018. Our study is based on an exhaustive review of the literature from FMIS representing 43 articles.

As we focus on the formal study of properties related to the graphical scene of interactive systems, this survey is based on a table of our choice that classifies the work that has been done about formalisation and verification of properties for interactive systems.

1.3 Plan of the article

Before reviewing the work from FMIS, we present our analytical framework (2). It is composed by definitions of properties we have sorted in different classes. We also set up a nomenclature of formalisms that have been used for the studies of the properties. From this basis, we propose an analytical grid that allows us to synthesize the review. Then the 43 articles from all the FMIS workshops are presented and analysed (3), analysis mainly directed by the studied properties and the ways of studying them.

The section 4 provides a synthesis of the review and highlights the issues in the research of formal methods for interactive systems. Finally, the section 5 concludes the discussion and presents ongoing work related to the previously highlighted issues.

2 Analytical framework

The purpose of this section is to define a framework for the analysis of the properties that have been studied for interactive systems. In order to do that, three basic questions must be considered.

- *"What properties are studied?"* This question concerns the nature of properties that have been studied and is the center of our work to determine if some properties have not been studied.

- *"What formalism is being used?"* This question allows us to show what formalisms can be used to study the properties.
- *"What is the case study?"* This question concerns the system used as the case study to illustrate the use of formal methods and its particularities.

We focus on these questions in order to highlight the range of interactive systems properties covered. It provides the means used to cover these properties. Through this survey, we explore these questions by sorting the articles by the properties studied and the means used to study those. We also provide the case study used to illustrate the studies.

2.1 Properties

As stated in the analytical framework, we firstly drive our analysis according to the studied properties. This paper organises interactive systems properties in four classes of our choice: user behavior [2], cognitive principles [29], human-machine interfaces [13], security [70]. We detail these classes below.

Several articles do not directly address interactive properties and so cannot be classified in one of these 4 classes. For these specific papers, we have defined two additional categories:

- **specification/formal definition**: gather papers dealing with the formal modeling of a system, and possibly addressing properties related to the model itself, and not centered on the interaction.
- **testing**: gather papers related to the modeling of interactive systems with the objective to generate test cases from the study of the model.

2.1.1 User behavior

This user behavior class considers the properties related to a human user. The properties from this class are about user's actions, user's expectations about the system, user's objectives and restrictions.

- A **user goal** is a list of sub objectives that a user has to perform to achieve a greater objective related to the purposes of the system used. This goal can consist on a single task or an overall use case.
  "Insert the card", "authenticate" and "choose the amount of money" are subgoals of "withdraw money".
- **User privileges** are a way to prevent a user with an unauthorized level of accreditation to perform goals the user should not.
  Example: It is only possible to access our e-mails if we are connected to our e-mail system.
- The **user interpretation** can be seen as the set of assumptions of the user about the system. It can lead users to adapt their behavior in accordance with these assumptions.
  For example, we are used to the shortcut Ctrl+C in order to copy some text. A novice user of a terminal could use it to copy text and close the running application because the functionality is not the same.
- The **user attention** is defined as the ability of the user to focus on a specific activity without being disturbed by irrelevant informations.
  This can be seen when driving a car, the driver is focused on traffic signs, on road traffic, etc.
- The **user experience** concerns the knowledge of the user about the system. This knowledge can come from a previous use of the system or a study of the system before using it. This experience can have an impact on user interpretation.
  The example given in user interpretation also illustrates the user experience: an experienced user of a terminal would not make mistakes with the Ctrl+C functionality.

2.1.2 Cognitive principles

This cognitive principles class considers the properties related to cognitive sciences. The properties from this class are about the human user cognitive salience and load.

- The cognitive **load** is related to the task performed by the user and more specifically to its complexity. It is possible to define two types of cognitive load: intrinsic (complexity of the task) and extraneous (complexity due to the context and distractors).
  For example, a user may lose attention while interacting with too rich a graphical scene.
- The cognitive **salience** represents a user's adherence to an idea. While performing an action, it depends on the action sensory salience, its procedural cueing and the cognitive load related to the task.
  A user will be more focused on an action more in line with his convictions.

2.1.3 Human-machine interfaces

In the human-machine interfaces properties class we consider the new properties that have arrived with these new systems. These properties are mainly specific to the problems induced by the display such as verifying the right display of informations or being aware of the latency that can appear between user actions and the display of informations.

- The **latency** is a well-known issue in rich interfaces. It concerns the delay between interactions with an application and the return of informations from it.
  If a computer processes several actions at the same time, it will take a few seconds to start a web browser.
- the **consistency** represents a system constant behavior whether for a display or a functionality regardless the current mode of the system.
  It can be seen as the use of same terminology for functions ("Exit" or "Quit" in order to define a function "close a window").
- The **predictability** is the user's ability to predict the future behavior of the system from its actual state and the way the user will interact with it.
  When closing a word processor with an unsaved document, a user knows that a pop-up will show to ask what to do between saving the document, canceling the closing or closing without saving.
- ISO 9241-11 [80] defines the **usability** 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."
  It is possible to improve the usability of an "accept/decline" window by adding symbols related to the two notions such as ✓for accept and $\times$ for decline.
- The visual **perceptibility** is based on different properties such as the superposition of components, the distinction of shapes and colours.
  For example, even if a red text is above a red shape, the text will not be perceptible.

2.1.4 Security

This security class considers the properties related to computer security such as the prevention of threats and the link between the user behavior and the possible threats.

- The **integrity** property states that, for a system that may be exposed to threats, hypothesis of the user about the application are correct and the reverse is also true.

When we log in interfaces with two text fields, if the fields username and password are not in the expected locations, we could type the password in the clear field.

- The **threats** property focuses in defining the differents threats that may be a risk for the system. We can note, for example, data leaking or data manipulation.

2.2 Nomenclature of formalisms

This section will introduce formalisms and formal methods that have been used by FMIS authors in order to formalize and apply verification techniques on the properties defined in the last section. We will define the basic semantic and the properties inherent to these formalisms.

2.2.1 Process algebra

Baeten [7] gives the history and the definition of process algebra. The author also gives examples of some formalisms from process algebra such as Calculus of Communicating Systems (CCS) or Communicating Sequential Processes (CSP). We can resume from this paper that process algebra is a set of algebraic means used to study and define the parallel systems behavior.

Authors from the FMIS workshop used formalisms from process algebra such as the CWB-NC [34] syntax for the Hoare's CSP notation [49], Language Of Temporal Ordering Specification (LOTOS) [52], probabilistic π-calculus [61], applied π-calculus [69], Performance Evaluation Process Algebra (PEPA) [48].

2.2.2 Specification language

A specification language [71] is a formal language that can be used to make formal descriptions of systems. It allows a user to analyze a system or its requirements and thus to improve its design.

Authors from the FMIS workshop used specification languages such as SAL [36], Z [79], μCharts [42], Spec# [11], Promela [50], PVS [74], Higher-Order Processes Specification (HOPS) [38].

2.2.3 Refinement

A program refinement consists in the concretisation of a more abstract description of a system. The aim of this method is to verify properties in an abstract level of the description then to concretise this level while conserving the verified properties. These steps have to be done until the concrete description of the system is obtained.

Authors from the FMIS workshop used refinements processes with models such as B-method [3] or with specification languages such as Z and μCharts.

2.2.4 Transition systems

Transition systems [8] consist in directed graphs composed of states, represented by nodes, and transitions, represented by edges. A state represents an instant in the system behavior or for a program the current value of all the variables and the current state of the program. Crossing a transition involves a change of state.

Authors from the FMIS workshop used transition systems formalisms such as UPPAAL [15], Petri nets (PN) [35], ICO models [60], finite state automata (FSA), Input/Output labeled transition system (IOLTS).

2.2.5 Temporal logic

Properties to be verified are often expressed in the form of temporal logic formulas [43]. These formulas are based on Boolean combiners, time combiners and for some logics on path quantifiers. Authors from the FMIS workshop used temporal logics such as computation tree logic (CTL) and linear temporal logic (LTL) [25].

3 Review

Here, we review the state of the art of formal methods applied to interactive systems. We consider research work that have been presented in the International Workshop on Formal Methods for Interactive Systems.

Our aim is to present the properties that have been studied with formal methods. From this and the questions that we asked in the section 2, we base our analysis on the grid presented in the table 6.

This grid highlights the coverage of properties depending on the formalisms. The categories formal definition/specification and testing are not interactive systems properties. However, we want to present how articles address those with formal methods. This explains the fact that there is a double vertical line in the grid. Our work addresses the visual perceptibility property from the HMI class. We highlight this by setting the perceptibility in italic beside the HMI class, separated by a dashed line.

3.1 User behavior

The table 1 summarizes the studies of the user behavior class of properties. It sorts the papers according to the properties studied (goals, privileges, interpretation, attention, emotion and experience) and formalisms used.

Table 1: Study of the **user behavior** class of properties in the FMIS workshops

	Goals	Privileges	Interpretation	Attention	Emotion	Experience
(PA) CWB-NC	[32]	[32]				
(PA) CSP	[30]			[30]		[30]
(PA) LOTOS				[81]		
(PA) PEPA						[33]
(SL) SAL	[65] [66]		[65] [67]			
(SL) HOPS	[37]					
(other) HTDL				[31]		[31]
ad-hoc formalism					[19]	

3.1.1 User goals

Cerone and Elbegbayan [32] define user goals in the use of a web-based interface that features a discussion forum and a member list. Those are defined with the CWB-NC syntax for CSP from process algebra. These definitions allow authors to model more precisely the attended and unattended use cases.

Rukšėnas et al. [65] address the use of an authentification interface with two textboxes (user name and password). They define user goals with the specification language SAL through the definition of a cognitive architecture of user behavior. It allows authors to define the actions a user can do. Rukšėnas et al. [66] further explore the notion of user goals through their cognitive architecture.

Cerone [30] bases his work on the study of two use cases: a driving user and a user interacting with an ATM. He models the user goals with the Hoare's notation for describing CSP (process algebra). It allows him to study cognitive activities such as closure.

Dittmar and Schachtschneider [37] use HOPS (specification language) models to define user tasks and actions while solving a puzzle.

3.1.2 User privileges

Cerone and Elbegbayan [32] define user privileges with the CWB-NC syntax for CSP. Thus, authors can model wich actions logged or non-logged users are allowed to do. This allows authors to constrain the user behavior by adding new properties in the web interface model.

3.1.3 User interpretation

Rukšėnas et al. [65] address the user interpretation of an authentification interface. They define it with SAL through the definition of a cognitive architecture of user behavior. It allows authors to highlight the risk for the user of misunderstanding the interface depending on the display of the two textboxes. Rukšėnas and Curzon [67] study the plausible behavior of users interacting with number entry on infusion pumps. They assume that users have their own beliefs about the incremental values. They separately model the users behavior depending on their interpretation and the constraint on cognitive mismatches with LTL and the SAL model checker.

3.1.4 User attention and user experience

Su et al. [81] study the temporal attentional limitation in the presence of stimuli on stimulus rich reactive interfaces. The cognitive model of human operators is defined with LOTOS (process algebra). The model of SRRI is based on studies of an AB task [40]. This work presents simulation results focusing on the performance of the interface in user attention.

Cerone [30] addresses user's expectations, which relies on user attention and user experience. He studies cognitive activities such as closure, contention scheduling and attention activation. He models those with the Hoare's notation for describing CSP (process algebra).

Cerone and Zhao [33] use the process algebra PEPA to model a three-way junction with no traffic lights and a traffic situation. They study the user experience in driving in such junctions. They use the PEPA Eclipse plug-in to analyse the model and determine for example the probability of possible collision.

Cerone [31] proposes a cognitive architecture for the modelling of human behavior. This work presents the Human Task Description Language (HTDL). He uses it to model properties related to user behavior such as the automatic (everyday tasks) and deliberate (driven by a goal) control and the human learning, attention and experience.

3.1.5 User emotion

Bonnefon et al. [19] use their logical framework, an ad-hoc formalism, to model several emotions and the notion of trust. Among the emotions there is joy/distress, hope/fear, satisfaction/disappointment and fear-confirmed/relief. They also model the relation between trust and emotions.

3.2 Cognitive principles

The table 2 summarizes the studies of the cognitive principles class of properties. It sorts the papers according to the properties studied (salience and load) and formalisms used.

Table 2: Study of the **cognitive principles** class of properties in the FMIS workshops

	Salience	Load
(SL) SAL	[66] [51]	[66] [51]
(other) GUM	[51]	[51]

Rukšėnas et al. [66] define two cognitive principles, salience and cognitive load. They add those to their SAL cognitive architecture. The authors also define the link between these two principles. They illustrate these principles through the case study of a Fire Engine Dispatch Task.

Huang et al. [51] try to see if their Generic User Model (GUM) can encapsulate all the cognitive principles presented in the Doughnut Machine Experiment [4].

3.3 Human machine interfaces

The table 3 summarizes the studies of the HMI class of properties. It sorts the papers according to the properties studied (consistency, predictability, latency and usability) and formalisms used.

Table 3: Study of the **HMI** class of properties in the FMIS workshops

	Consistency	Predictability	Latency	Usability	Perceptibility
(SL) SAL		[57]		[65] [66]	
(SL) PVS	[46] [47]				
(SL/Re) μCharts				[22]	
(SL/Re) Z	[21]				
(TS) IOLTS	[14]				
(TL) LTL	[14]			[66]	
(TL) CTL	[46] [27]				
(other) Tree based WCET			[55]		

3.3.1 Consistency

Bowen and Reeves [21] use their presentation models and refinement processes with Z to check the equivalence and the consistency between two UI designs. The presentation models allow them to ensure that controls with the same function have the same name and conversely.

Beckert and Beuster [14] provide an IOLTS model of a text-based application to guarantee consistency constraints. Their first model does not satisfy consistency constraints. They refine this model in order to satisfy the consistency constraints.

Campos and Harrison [27] provide consistency a formal definition of consistency of the Alaris GP Volumetric Pump interface in CTL. The global consistency includes: the role and visibility of modes, the relation between naming and purpose of functions, consistency of behavior of the data entry keys. They also present a part of a MAL specification of the Alarais GP infusion pump.

Harrison et al. [46] explore the consistency in the use of the soft function keys of infusion pumps through the use of MAL models translated into PVS. They define consistency properties with CTL and translate those into PVS theorems.

Harrison et al. [47] create a model of a pill dispenser from a specification in PVS. They use this specification with the PVSio-web tool to study the consistency of possible actions.

3.3.2 Predictability

Masci et al. [57] analyse the predictability of the number entry system of Alaris GP and B-Braun Infusomat Space infusion pumps. They use SAL specifications to specify the predictability of the B-Braun number entry system.

3.3.3 Latency

Leriche et al. [55] explore the possibility of using Worst-Case Execution-Time [64] based on trees to study the latency for interactive systems. They also present some works that have been done with graphs of activation to model interactive systems.

3.3.4 Usability

Rukšėnas et al. [65] use their user behavior model in SAL to check usability properties of an authentification interface. They check that the property "the user eventually achieves the perceived goal" is satisfied. Rukšėnas et al. [66] further explore the use of their user model with SAL and LTL properties. They check that the property "the user eventually achieves the main goal" is satisfied in the Fire Engine Dispatch Task.

Bowen and Reeves [22] present a way of applying the specification language μCharts and refinement processes to UI designs. They use presentation models to compare two UI designs and if these UI maintain usability. They also informally describe the refinement process related to UI design.

3.4 Security

The table 4 summarizes the studies of security class of properties. It sorts the papers according to the properties studied (integrity, usability errors and threats) and formalisms used.

Table 4: Study of the **security** class of properties in the FMIS workshops

	Integrity	Usability errors	Threats
(SL) SAL		[65]	
(TS) IOLTS	[14]		
(TL) LTL	[14]		
(other) BDMP			[53]
others/ad-hoc	[6]	[6]	

Rukšėnas et al. [65] check the risk of security breach in the authentification interface with SAL properties. This highlights the fact that user interpretation can impact the security by entering the password in the wrong textbox for example.

Beckert and Beuster [14] produce a generic IOLTS (transition system) model of a text-based application. They use LTL to describe the properties of components and interpret them with IOLTS. The model is refined to guarantee integrity and to consider the problem of multi-input (if the user enters again a data if the system has not yet processed the last one) risking security breaches.

Arapinis et al. [6] present security properties related to the use of the MATCH (Mobilising Advanced Technology for Care at Home) food delivery system. They define these properties by using different formalisms such as the access control language RW and temporal logic (LTL, TCTL, PCTL).

Johnson [53] studies security properties in terms of threats that may occur on Global Navigation Satellite Systems (GNSS). He models GNSS with Boolean Driven Markov Processes (BDMP) and integrate security threats to the model.

3.5 Specification/formal definition and testing

The table 5 summarizes the studies of the specification/formal definition and testing classes. It sorts the papers according to the case (specification/formal definition and testing) and formalisms used. This section allows us to present different systems used as case studies.

The references concern the articles that address the formal definition or specification of systems. These articles do not cover the properties previously presented. We only present in this section these articles.

Table 5: Study of the **specification/formal definition** and **testing** classes in the FMIS workshops

	Formal definition	Testing
(PA) CSP	[30]	
(PA) LOTOS	**[10]**	
(PA) π-calculus	[6]	
(PA) Prob. π-calc.	**[5]**	
(PA) PEPA	[33]	
(PA) TCBS'	**[16]**	
(SL) SAL	[57] **[12]**	
(SL) Spec#		[75]
(SL) PVS	**[56]** [46] **[62]** [47]	
(SL) Promela	**[26]**	
(SL) HOPS	[37]	
(SL/Re) μCharts	[22]	
(SL/Re) Z	[21] [23]	[23]
(Re) B/event-B	**[28] [68] [41]**	
(TS) FSM	**[82]**	
(TS) UPPAAL	**[45]**	
(TS) Colored PN	**[76]**	

	Formal definition	Testing
(TS) GTS	**[84]**	
(TS) FSA	**[83]**	
(TS) Event act. graph	[55]	
(TS) ICO	**[76]**	
(TL) LTL	[6] **[26]**	
(TL) CTL	[46]	
(other) SAT	**[26]**	
(other) Mark. proc.	**[5]**	
(other) MAL	[27] [46]	
(other) GUM	[51]	
(other) BDMP	[53]	
(tool) Spec explorer		[75]
(tool) FEST		[75]
(tool) SMT solver		[23]
(tool) PVSio web	**[62]** [47]	
others / ad-hoc	[77] [17] [39] [6] [30] **[20] [82] [9]**	

3.5.1 Specification/formal definition

We sort the articles only focused in specification/formal definition by formalism used.

Process algebra Barbosa et al. [10] represent an air traffic control system with a control tower and three aircrafts as CNUCE interactors. They use ad-hoc formalism, a generic approach to process algebra, to define this representation.

Anderson and Ciobanu [5] builds a Markov Decision Process abstraction of a program specification expressed with a probabilistic process algebra (using π-calculus). The abstraction is then used to check the structure of specification, analyze the long-term stability of the system, and provide guidance to improve the specifications if they are found to be unstable.

Bhandal et al. [16] present the language TCBS', strongly based on the Timed Calculus of Broadcasting Systems (TCBS). They give a formal model of a coordination model, the Comhordú system, in this language.

Specification language Calder et al. [26] study the MATCH Activity Monitor (MAM), an event driven rule-based pervasive system. They model separately the system behavior and its configuration (rule set) with Promela. They derive Promela rules in LTL properties to check redundant rule with the model checker SPIN.

Bowen and Hinze [20] present early stages work using presentation models to design a tourist information system. This system displays a map on a mobile support (smartphone).

Bass et al. [12] specify in SAL the three subsystems of the A320 Speed Protection: automation, airplane and pilots. This interactive hybrid system has the potential to provide an automation surprise to a user.

Masci et al. [56] specify the DiCoT's information flow model by using PVS. They use three modelling concepts (system state, activities, task) for this specification. The authors use the example of the London Ambulance Service to illustrate their work.

Refinement Cansell et al. [28] specify an interface of e-voting corresponding to the Single Transferable Vote model without the counting algorithm. This is done by using the B method and a refinement process.

Rukšėnas et al. [68] study the global requirements related to data entry interfaces of infusion pumps. They use Event-B specifications and refinement processes with the Rodin platform to specify these requirements. These refinement processes allow the authors to check if the Alaris GB infusion pump number entry specification validate the global requirements.

Geniet and Singh [41] study an HMI composed by graphical components in form of widgets. They use the Event-B modelling language and refinement processes to model the system and analyse its behavior.

Transition system Harrison et al. [45] model the GAUDI system [54] with UPPAAL. Through the UPPAAL model, the authors can explore use cases scenario and check reachability properties for example.

Westergaard [84] uses game transition systems to define visualisations of the behavior of formal models. The example of an interoperability protocol for mobile ad-hoc networks to highlight the use of visualisations.

Thimbleby and Gimblett [82] model the interactions possibilities with key data entry of infusion pumps. They use FSM and specify those with regular expressions to model the interactions.

Silva et al. [76] formally define a system and its WIMP and Post-WIMP interactions with ICO models and colored Petri nets. These models allow them to analyse the properties inherent to the formalisms: place transitions invariants, liveness and fairness, and reachability.

Turner et al. [83] generate presentation models describing tasks and widgets based interactions sequences of an infusion pump. It is composed by five buttons (Up, Down, YesStart, NoStop, OnOff) and a display allowing interactions with the user. They use FSA to model these sequences.

Others Bhattacharya et al. [17] model soft keyboards (on-screen keyboards) with scanning and use the Fitts-Digraph model [78] to evaluate the performance of their model and the system.

Sinnig et al. [77] describe a new formalism based on sets of partially ordered sets. They use it to formally define use cases and task models.

Dix et al. [39] use an ad-hoc formalism to model physical devices (switches, electric kettle, etc.) logical states and their digital effects in another model.

Oladimeji et al. [62] present PVSio-web, a tool which extends the PVSio component of PVS with a graphical environment. They demonstrate its use by prototyping the data entry system of infusion pumps.

Banach et al. [9] consider using an Event-B model in conjunction with an SMT solver in order to proof some invariants on a hardware based components, dedicated to the acquisition and fusion of inputs from various sensors to a visually impaired and blind person's white cane (INSPEX project).

3.5.2 Testing

Silva et al. [75] highlight a way of testing model-based graphical user interfaces. The testing process presented is as follows: a FSM model called Presentation Task Sets (PTS) is generated from a task model (CTT) with the TERESA tool [58], a Spec# oracle is generated from the FSM model with their Task to Oracle Mapping (TOM) tool, then a testing framework is used to test the system against the oracle.

Bowen and Reeves [23] use the specification language Z for specifying a calendar application. They explore the way to apply testing processes on this application. They use their presentation and interaction models to derive tests such as ensuring that the relevant widgets exist in the appropriate states and ensuring that the widgets have the required behaviors.

4 Findings

Through this survey, we have explored the study of interactive systems with formal methods. Several classes of properties have been studied and cover different aspects of interactions.

The table 6 summarizes the studies of the articles from the International Workshop on Formal Methods for Interactive Systems that has taken place seven times from 2006 to 2018. It gives a distribution of the articles in our analytical grid. We note: ✓: 1-5 articles; ✓✓: 6-10 articles; ✓✓✓: 10+ articles.

Table 6: Study of interactive systems properties in the FMIS workshops

	User behavior	Cognitive pr.	HMI Perceptibility	HMI Others	Security	Formal def.	Testing
Process algebra	✓					✓✓	
Spec. language	✓	✓		✓✓	✓	✓✓✓	✓
Refinement				✓		✓	✓
Transition systems	✓			✓	✓	✓✓	
Temporal logic		✓		✓	✓	✓	
Other / ad-hoc	✓	✓		✓	✓	✓✓	✓

High proportion of works on formal definitions and specifications

We highlight the high proportion of articles that address the formal definition and the specification of interactive systems (classified in "Formal def." column of table 6). Among the 43 articles from the FMIS workshops, 34 are related to this aspect (representing approximately 80%). More than the half of those specifically address the formal definition of properties inherent to the formalisms used (invariant for B, reachability for transition systems, etc.).

Perceptibility unstudied

We can note that even if several properties related to HMI have been studied, no paper addresses perceptibility properties (cf. "Perceptibility" column). In the FMIS workshops, we have not spotted studies addressing visual, sound or haptic based interactions.

Common formalisms

If we look at the formalisms used (table 5), it appears that some are in the majority.

We can see that PVS and SAL are the most widely used specification languages. Over the 14 articles that use specification languages, we find that SAL is the most used with 5 articles using it. PVS is also widely used with 4 articles using it. Those two cover more than the half of the articles using specification language.

B and event-B models are still the most used for refinement processes. 6 articles present refinement processes and half of those use B and event-B models. We find 2 articles using Z and 1 article using μCharts.

New formalisms

During this analysis, we have seen some formalisms close to the nomenclature we have set (see section 2.2). But other formalisms could not be easyly classified in one of the proposed families. We identified 8 papers that use ad-hoc formalisms or formalisms out of the nomenclature.

In those we find, for example, the formal definition of task models and use cases by using an ad-hoc formalism based on sets of partially ordered sets. We also find the modelling of several physical devices with a new and ad-hoc formalism. Another paper presents the formal definition of different emotions by using an ad-hoc formalism. An article presents security properties and the different means (access control language RW, ProVerif's query language, applied π-calculus) of formalising those.

Maturity of case studies

A main case study is frequently presented: the "infusion pump" system. Other systems are presented and considered as "textbook" cases, representing more than half of the papers.

The infusion pump is a safety critical medical device and is used by 7 out of 43 articles. 3 of those study the data part of the whole system by modelling it and validate some properties on a sub-system only. 3 other articles study the full system. They model the final device or its specification in order to check whether the device or its specification validate the global requirements of infusion pump. The last article studies the possible interactions between a user and the system. They model those in the form of interaction sequences corresponding to the human user tasks.

This approach demonstrated the feasibility of the proposed methods but remains limited. We note that even if an infusion pump is a safety critical system, the studies made for this system do not necessarily address safety critical aspects. Indeed, only 3 articles focus on the full system and its certification oriented requirements. Only those demonstrate the scalability of the formalisms used.

16 out of 43 articles focus on "textbook" cases and address the user interface part (web application, smartphone application, e-voting system, etc.). Those allow authors to easily illustrate the use of several formal methods and the properties inherent to those. The systems are modelled, several properties, inherent to the formalisms or to the systems, are studied. However, these articles only illustrate the formal methods and do not allow authors to demonstrate the potential scalability of these formal methods.

5 Conclusion

Aim and contribution of this article

The aim of this article is to review different research work on formal methods applied to interactive systems. The overall contribution is to provide a review of the literature, 43 articles, from the International Workshop on Formal Methods for Interactive Systems. This workshop took place seven times from 2006 to 2018. First we propose an analytical framework based on a few questions. Then we present several properties of interactive systems and classify them. We set a classic nomenclature of formalisms. This analytical framework is provided with an analysis grid of our own. Those highlight the following points: formalisms used, properties studied, case study used to illustrate the analysis. Finally, we highlight the findings and the outgoing issues.

Discussion

Interactive systems are increasingly used in several sectors and propose several kinds of interactions with human users. The interactions can be from the system to the user by using sound notifications or display notifications in order to provide information to the user about the actual internal state of the system. They can also be from the user to the system with many interaction solutions such as mouse clicks, data

entries with keyboards or buttons on the system or soft keyboards and buttons on the display of the system interface. All these interactions are source of new challenges when when the objective is to perform the formal verification and validation of their related properties.

During the last decade, a substantial work has been done in order to study how formalisms and methods can be applied to interactive system. A lot of them have demonstrated that it is possible to take into account a lot of classes of properties. High level properties such as those related to the tasks the user may accomplish or those related to the abstract interface have been studied. The classical formalisms relying on state and transition paradigm can be easily used to model these elements. However, properties related to the concrete part of the interface (involving characteristics of the graphical scene) remain largely uncovered by studies. As we highlighted in the section 4, we note that the properties related to the perceptibility have not yet been studied. This is not a real surprise: these properties require to model characteristics of the system which are not traditionaly handled by formal models: color of graphical objects, forms, dimension, visibility, collision etc. Modeling them remains a big challenge.

Perspectives

Our research team works in the aeronautics sector. Thus, we focus on interactive and critical systems related to this sector. Interfaces with a very rich graphical scene are becoming increasingly important in aircraft cockpits. In this context, we develop a reactive language, Smala[1], allowing us to develop interfaces and interactions at the same level.

The issue related to visual perceptibility properties is then important in our opinion. In Béger et al. [24] we propose elements for formalising graphical properties. We set three basic properties that compose the node of our formalism: the display order depending on the display layer of graphical elements, the intersection depending on the domain of graphical elements and the colour equality. We also present a scene graph we extract from the Smala source code. It models interactive systems and their graphical interface in a new way. It also gives information about graphical elements and their variables (position, colour, opacity, etc.).

From those, we can formally define graphical requirements for an aeronautic system specified in a standard (ED-143 [1]). The formalism defines requirements such as the colour equality/inequality, the authorized/unauthorized positions and the display order. The scene graph defines requirements we can not write with our formalism such as the shape of graphical elements.

We aim at defining new graphical properties in order to express with our formalism requirements related to the shape and the relative positions of graphical elements. In order to automatically validate the requirements, we want to link our formalism to the Smala source code by using code annotations.

Acknowledgments

This work is partly funded by the ANR project FORMEDICIS, ANR-16-CE25-0007.

References

1. Ed 143 - minimum operational performance standards for traffic alert and collision avoidance system ii (tcas ii), April 2013.

2. John A Bargh. *The Four Horsemen of Automaticity: Awareness, Efficiency, Intention, and Control in Social Cognition.*, volume 2. 01 1994.

3. J.-R. Abrial. *The B-book: Assigning Programs to Meanings*. Cambridge University Press, New York, NY, USA, 1996. ISBN 0-521-49619-5.

[1] `http://smala.io/`

4. MGA Ament, Anna Cox, Ann Blandford, and Duncan Brumby. Working memory load affects device-specific but not task-specific error rate. *CogSci 2010: Proceedings of the Annual Conference of the Cognitive Science Society*, pages 91 – 96, 2010.

5. Hugh Anderson and Gabriel Ciobanu. Markov abstractions for probabilistic pi-calculus. *Electronic Communications of the EASST*, 22, 01 2009. ISSN 1863-2122.

6. Myrtho Arapinis, Muffy Calder, Louise Denis, Michael Fisher, Philip Gray, Savas Konur, Alice Miller, Eike Ritter, Mark Ryan, Sven Schewe, Chris Unsworth, and Rehana Yasmin. Towards the verification of pervasive systems. *Electronic Communications of the EASST*, 22, 01 2009. ISSN 1863-2122.

7. J.C.M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2):131 – 146, 2005. ISSN 0304-3975. Process Algebra.

8. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008. ISBN 026202649X, 9780262026499.

9. Richard Banach, Josph Razavi, Olivier Debicki, Nicolas Mareau, Suzanne Lesecq, and Julie Foucault. Application of formal methods in the inspex smart systems integration project. In *FMIS 2018*, 5 2018.

10. Marco Antonio Barbosa, Luís Soares Barbosa, and José Creissac Campos. Towards a coordination model for interactive systems. *Electronic Notes in Theoretical Computer Science*, 183:89 – 103, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

11. Mike Barnett and and. The spec# programming system: An overview. In *CASSIS 2004, Construction and Analysis of Safe, Secure and Interoperable Smart devices*, volume 3362 of *Lecture Notes in Computer Science*, pages 49–69. Springer, January 2005.

12. Ellen J. Bass, Karen M. Feigh, Elsa Gunter, and John Rushby. Formal modeling and analysis for interactive hybrid systems. 45, 01 2011.

13. Michel Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 15–22, New York, NY, USA, 2004. ACM. ISBN 1-58113-867-9.

14. Bernhard Beckert and Gerd Beuster. Guaranteeing consistency in text-based human-computer-interaction. 2007. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

15. Gerd Behrmann, Alexandre David, and Kim G. Larsen. *A Tutorial on Uppaal*, pages 200–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-30080-9. doi: 10.1007/978-3-540-30080-9_7.

16. Colm Bhandal, Melanie Bouroche, and Arthur Hughes. A process algebraic description of a temporal wireless network protocol. 45, 01 2011.

17. Samit Bhattacharya, Anupam Basu, Debasis Samanta, Souvik Bhattacherjee, and Animesh Srivatava. Some issues in modeling the performance of soft keyboards with scanning. 2007. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

18. Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Formalization of Real Analysis: A Survey of Proof Assistants and Libraries. *Mathematical Structures in Computer Science*, 26(7):1196–1233, October 2016.

19. Jean-François Bonnefon, Dominique Longin, and Manh Hung Nguyen. A logical framework for trust-related emotions. *Electronic Communications of the EASST*, 22, 01 2009. ISSN 1863-2122.

20. Judy Bowen and Annika Hinze. Supporting mobile application development with model-driven emulation. 45, 01 2011.

21. Judy Bowen and Steve Reeves. Formal models for informal gui designs. *Electronic Notes in Theoretical Computer Science*, 183:57 – 72, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

22. Judy Bowen and Steve Reeves. Refinement for user interface designs. *Electronic Notes in Theoretical Computer Science*, 208: 5 – 22, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

23. Judy Bowen and Steve Reeves. Ui-design driven model-based testing. *Electronic Communications of the EASST*, 22, 01 2009. ISSN 1863-2122.

24. Pascal Béger, Valentin Becquet, Sébastien Leriche, and Daniel Prun. Contribution à la formalisation des propriétés graphiques des systèmes interactifs pour la validation automatique. In *Afadl 2019, 18èmes journées Approches Formelles dans l'Assistance au Developpement de Logiciels* , Toulouse, France, June 2019.

25. B. Bérard et al. *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 3642074782, 9783642074783.

26. Muffy Calder, Phil Gray, and Chris Unsworth. Tightly coupled verification of pervasive systems. *Electronic Communications of the EASST*, 22, 01 2009. ISSN 1863-2122.

27. José Campos and Michael Harrison. Modelling and analysing the interactive behaviour of an infusion pump. 45, 01 2011.

28. Dominique Cansell, J. Paul Gibson, and Dominique Méry. Refinement: A constructive approach to formal software design for a secure e-voting interface. *Electronic Notes in Theoretical Computer Science*, 183:39 – 55, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

29. Ula Cartwright-Finch and Nilli Lavie. The role of perceptual load in inattentional blindness. *Cognition*, 102(3):321 – 340, 2007. ISSN 0010-0277.

30. Antonio Cerone. Closure and attention activation in human automatic behaviour: A framework for the formal analysis of interactive systems. 45, 01 2011.

31. Antonio Cerone. Towards a cognitive architecture for the formal analysis of human behaviour and learning. In Manuel Mazzara, Iulian Ober, and Gwen Salaün, editors, *Software Technologies: Applications and Foundations*, pages 216–232, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04771-9.

32. Antonio Cerone and Norzima Elbegbayan. Model-checking driven design of interactive systems. *Electronic Notes in Theoretical Computer Science*, 183:3 – 20, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

33. Antonio Cerone and Yishi Zhao. Stochastic modelling and analysis of driver behaviour. *ECEASST*, 69, 2013.

34. Rance Cleaveland, Tan Li, and Steve Sims. The concurrency workbench of the new century. User's manual, SUNY at Stony Brook, Stony Brooke, NY, USA, 2000.

35. R. David and Hassane Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer Publishing Company, Incorporated, 2nd edition, 2010. ISBN 3642106684, 9783642106682.

36. Leonardo de Moura, Sam Owre, Harald Rueß, John Rushby, N. Shankar, Maria Sorea, and Ashish Tiwari. Sal 2. In Rajeev Alur and Doron A. Peled, editors, *Computer Aided Verification*, pages 496–500, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-27813-9.

37. Anke Dittmar and Reik Schachtschneider. Lightweight interaction modeling in evolutionary prototyping. *ECEASST*, 69, 2013.

38. Anke Dittmar, Toralf Hübner, and Peter Forbrig. Hops: A prototypical specification tool for interactive systems. In T. C. Nicholas Graham and Philippe Palanque, editors, *Interactive Systems. Design, Specification, and Verification*, pages 58–71, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-70569-7.

39. Alan Dix, Masitah Ghazali, and Devina Ramduny-Ellis. Modelling devices for natural interaction. *Electronic Notes in Theoretical Computer Science*, 208:23 – 40, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

40. Jane E. Raymond, Kimron Shapiro, and Karen Arnell. Temporary suppression of visual processing in an rsvp task: An attentional blink? *Journal of experimental psychology. Human perception and performance*, 18:849–60, 09 1992.

41. Romain Geniet and Neeraj Kumar Singh. Refinement based formal development of human-machine interface. In Manuel Mazzara, Iulian Ober, and Gwen Salaün, editors, *Software Technologies: Applications and Foundations*, pages 240–256, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04771-9.

42. Doug Goldson, Greg Reeve, and Steve Reeves. μ-chart-based specification and refinement. In *Proceedings of the 4th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering*, ICFEM '02, pages 323–334, Berlin, Heidelberg, 2002. Springer-Verlag. ISBN 3-540-00029-1.

43. Valentin Goranko and Antony Galton. Temporal logic. *The Stanford Encyclopedia of Philosophy* (Winter 2015 Edition), Edward N. Zalta (ed.), 2015.

44. Anjana Gosain and Ganga Sharma. Static analysis: A survey of techniques and tools. In *Intelligent Computing and Applications*, pages 581–591, New Delhi, 2015. Springer India. ISBN 978-81-322-2268-2.

45. Michael D. Harrison, Christian Kray, and José Creissac Campos. Exploring an option space to engineer a ubiquitous computing system. *Electronic Notes in Theoretical Computer Science*, 208:41 – 55, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

46. Michael D. Harrison, Paolo Masci, Jose Creissac Campos, and Paul Curzon. Automated theorem proving for the systematic analysis of an infusion pump. *ECEASST*, 69, 2013.

47. Michael D. Harrison, Paolo Masci, and José Creissac Campos. Formal modelling as a component of user centred design. In Manuel Mazzara, Iulian Ober, and Gwen Salaün, editors, *Software Technologies: Applications and Foundations*, pages 274–289, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04771-9.

48. Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, New York, NY, USA, 1996. ISBN 0-521-57189-8.

49. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985. ISBN 0-13-153271-5.

50. Gerard Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 1st edition, 2011. ISBN 0321773713, 9780321773715.

51. Huayi Huang, Rimvydas Rukšėnas, Maartje Ament, Paul Curzon, Anna Cox, Ann Blandford, and Duncan Brumby. Capturing the distinction between task and device errors in a formal model of user behaviour. 45, 01 2011.

52. ISO-8807:1989. Information processing systems - open systems interconnection - lotos - a formal description technique based on the temporal ordering of observational behaviour, 1989.

53. Chris W. Johnson. Using assurance cases and boolean logic driven markov processes to formalise cyber security concerns for safety-critical interaction with global navigation satellite systems. 45, 01 2011.

54. Christian Kray, Gerd Kortuem, and Antonio Krüger. Adaptive navigation support with public displays. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, pages 326–328, New York, NY, USA, 2005. ACM. ISBN 1-58113-894-6.

55. Sébastien Leriche, Stéphane Conversy, Célia Picard, Daniel Prun, and Mathieu Magnaudet. Towards handling latency in interactive software. In Manuel Mazzara, Iulian Ober, and Gwen Salaün, editors, *Software Technologies: Applications and Foundations*, pages 233–239, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04771-9.

56. Pablo Masci, Paul Curzon, Ann Blandford, and Dominic Furniss. Modelling distributed cognition systems in pvs. 45, 01 2011.

57. Paolo Masci, Rimvydas Rukšėnas, Patrick Oladimeji, Abigail Cauchi, Andy Gimblett, Yunqiu Li, Paul Curzon, and Harold Thimbleby. On formalising interactive number entry on infusion pumps. 45, 01 2011.

58. G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8):507–520, Aug 2004. ISSN 0098-5589. doi: 10.1109/TSE.2004.40.

59. Brad A. Myers and Mary Beth Rosson. Survey on user interface programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 195–202, New York, NY, USA, 1992. ACM. ISBN 0-89791-513-5.

60. David Navarre, Philippe Palanque, Jean-Francois Ladry, and Eric Barboni. Icos: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Trans. Comput.-Hum. Interact.*, 16(4):18:1–18:56, November 2009. ISSN 1073-0516.

61. G. Norman, C. Palamidessi, D. Parker, and P. Wu. Model checking the probabilistic π-calculus. In *Proc. 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, pages 169–178. IEEE Computer Society, 2007.

62. Patrick Oladimeji, Paolo Masci, Paul Curzon, and Harold Thimbleby. Pvsio-web: a tool for rapid prototyping device user interfaces in pvs. *ECEASST*, 69, 2013.

63. Susan Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4 (3):455–495, July 1982. ISSN 0164-0925.

64. Peter Puschner and Alan Burns. A review of worst-case execution-time analyses. *Real-time Systems - RTS*, 01 1999.

65. R. Rukšėnas, P. Curzon, and A. Blandford. Detecting cognitive causes of confidentiality leaks. *Electronic Notes in Theoretical Computer Science*, 183:21 – 38, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

66. R. Rukšėnas, J. Back, P. Curzon, and A. Blandford. Formal modelling of salience and cognitive load. *Electronic Notes in Theoretical Computer Science*, 208:57 – 75, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

67. Rimvydas Rukšėnas and Paul Curzon. Abstract models and cognitive mismatch in formal verification. 45, 01 2011.

68. Rimvydas Rukšėnas, Paolo Masci, Michael D. Harrison, and Paul Curzon. Developing and verifying user interface requirements for infusion pumps: A refinement approach. *ECEASST*, 69, 2013.

69. Mark D. Ryan and Ben Smyth. Applied pi calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, chapter 6. IOS Press, 2011.

70. A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE J.Sel. A. Commun.*, 21(1):5–19, September 2006. ISSN 0733-8716.

71. Donald Sannella and Martin Wirsing. Specification languages. *Algebraic Foundation of Systems Specification. IFIP State-of-the-Art Reports, Berlin: Springer, 1999, 243-272.*, 07 1999.

72. RTCA (Firm). SC-205 and EUROCAE (Agency). Working Group 71. Rtca/do-178c software considerations in airborne systems and equipment certification, December 2011.

73. RTCA (Firm). SC-205 and EUROCAE (Agency). Working Group 71. Rtca/do-333 formal methods supplement to do-178c and do-278a, December 2011.

74. N. Shankar. Pvs: Combining specification, proof checking, and model checking. In Mandayam Srivas and Albert Camilleri, editors, *Formal Methods in Computer-Aided Design*, pages 257–264, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-49567-3.

75. José L. Silva, José Creissac Campos, and Ana C.R. Paiva. Model-based user interface testing with spec explorer and concurtasktrees. *Electronic Notes in Theoretical Computer Science*, 208:77 – 93, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

76. José-Luis Silva, Camille Fayollas, Arnaud Hamon, Philippe Palanque, Célia Martiinie, and Eric Barboni. Analysis of wimp and post wimp interactive systems based on formal specification. *ECEASST*, 69, 2013.

77. Daniel Sinnig, Patrice Chalin, and Ferhat Khendek. Towards a common semantic foundation for use cases and task models. *Electronic Notes in Theoretical Computer Science*, 183:73 – 88, 2007. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

78. R. William Soukoreff and I. Scott Mackenzie. Theoretical upper and lower bounds on typing speed using a stylus and a soft keyboard. *Behaviour & Information Technology*, 14(6):370–379, 1995.

79. J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. ISBN 0-13-983768-X.

80. I.O.F. Standardization. *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*. 1998.

81. Li Su, Howard Bowman, and Philip Barnard. Performance of reactive interfaces in stimulus rich environments, applying formal methods and cognitive frameworks. *Electronic Notes in Theoretical Computer Science*, 208:95 – 111, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

82. Harold Thimbleby and Andy Gimblett. Dependable keyed data entry for interactive systems. 45, 01 2011.

83. Jessica Turner, Judy Bowen, and Steve Reeves. *Using Abstraction with Interaction Sequences for Interactive System Modelling: STAF 2018 Collocated Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers*, pages 257–273. 06 2018. ISBN 978-3-030-04770-2.

84. Michael Westergaard. A game-theoretic approach to behavioural visualisation. *Electronic Notes in Theoretical Computer Science*, 208:113 – 129, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.