Formal Description Techniques to Support the Design, Construction and Evaluation of Fusion Engines for SURE (Safe, Usable, Reliable and Evolvable) Multimodal Interfaces

Jean-François Ladry
IHCS - IRIT
University of Toulouse, France
ladry@irit.fr

David Navarre
IHCS - IRIT
University of Toulouse, France
navarre@irit.fr

Philippe Palanque
IHCS - IRIT
University of Toulouse, France
palanque@irit.fr

ABSTRACT

Representing the behaviour of multimodal interactive systems in a complete, concise and non-ambiguous way is still a challenge for formal description techniques (FDT). Depending on the FDT, multimodal interactive systems feature specific characteristics that are either cumbersome or impossible to capture with classical FDT. This is due to the multiple (potentially synergistic) use of modalities and the strong temporal constraints usually encountered in this kind of systems that have to be dealt with exhaustively if FDT are used. This paper focuses on the requirements for the modelling and construction of fusion engines for multimodal interfaces. It proposes a formal description technique dedicated to the engineering of interactive multimodal systems able to address the challenges of fusion engines. Such benefits are presented on a set of examples illustrating both the constructs and the process.

Categories and Subject Descriptors

D.2.2 **[Software]** Design Tools and Techniques - *Computer-aided software engineering (CASE),* H.5 **[Information Systems]** Information Interfaces and Presentation (e.g., HCI)

General Terms: Human factors, Reliability.

Keywords: Model-based approaches, formal description techniques, fusion engines, multimodal interfaces, interactive software engineering, safety-critical interactive systems

1. INTRODUCTION

The academic world has been providing prototypes, toolkits and toy systems offering multimodal interaction techniques since the early work from Bolt in the early 80's [8]. However, the actual engineering of multimodal interactive systems remains a cumbersome task, as it adds complexity to the design, specification, validation and implementation of interactive systems which is already a difficult task not addressed by current software engineering practice.

As model-based approaches already bring many advantages for the non-interactive part of a software system, it intuitively seems natural that extending these approaches can provide support for a more systematic development of multimodal interactive systems.

While designing user interfaces for the command and control of safety critical systems the designers face a difficult dilemma: either to stay with poor interaction techniques (but the system will be easier to design, implement, validate and certify) or to explore new interaction techniques in order to increase the bandwidth between users and system. In the latter, the development process will be more difficult (resources consumption will increase throughout the design, specification, validation and certification stages) but the use phase of the system is expected to improve effectiveness, efficiency and satisfaction of the users.

An important number of studies have been investigating the impact of using multimodal interaction techniques in safety critical systems domains:

- Military and civil aviation systems such as mission planning [27], [13], or interactive cockpits for civil [1] or military aircrafts [9], air traffic management with incident management [33],
- Crisis management system [21] for dispatching emergency vehicles,
- Interactive simulation training environments [10] in military systems such as SIMNET or Leathernet,
- Healthcare Medical informatics as in [10].

These previous studies (and additional ones such as [30], [10]) have been proposing and testing the use of multimodal interaction techniques in the field of safety critical systems have identified and reported several advantages:

- Multimodality increases reliability of the interaction. Indeed, it permits to decrease drastically critical error (between 36% and 50%) during interaction. This advantage alone can justify use of multimodality when interacting with a safety critical system.
- It increases the efficiency of the interaction, in particular in the field of spatial commands (multimodal interaction is 10% more rapid than classical interaction to specify geometric and localization information).
- Users predominantly prefer interacting in a multimodal way, probably because it allows more flexibility in interaction thus taking into account users' variability (especially if equivalence is provided).
- Multimodality allows increasing naturalness and flexibility of interaction so that learning period is shorter.

For all these reasons, multimodality has acquired greater interest in the field of command and control interactive systems. In particular, the last point is critical in that domain, as users of these systems are trained extensively to use them according to rules and regulations and to ensure that the operations will be performed in a safe and reliable manner.

Next section present the related work in the field of multimodal interfaces with a special highlight on fusion engines modelling. Section 3 presents how fusion engines can be modelled using a formal description technique and describes the benefits of employing such techniques. Section 4 presents a set of limitations of the approach introduced in section 3 and identifies, from that, a research agenda for future work.

2. RELATED WORK

Formal methods (thanks to their mathematical foundations) are typically used to describe and develop efficient, reliable, and safe systems [17]. They can be used in the various stages of the development process e.g. requirements, specifications … but they provide additional benefits by means of analysis and verification of models.

Using models for the design of computer systems provide a description of the system independent from its implementation. Nowadays such approaches are prominent in the area of software engineering via the Model Driven Engineering [25] field that emerged from the UML standard [29]. Indeed, as they provide a more abstract description of the system than the implementation code they also provide a unique opportunity for various stakeholders (designers, users, developers …) to comment and

Modelling fusion engines deals with both modelling interactive systems and modelling particular aspects of multimodality. The following three sections present the related literature.

2.1 Modelling Interactive Systems

In the area of interactive systems, models have, in addition, to be able to describe how user actions on input devices influence the behaviour of the system and how system's states are presented to the users by means of the output devices. Another important characteristic is that "recent" interactive systems are by nature reactive systems i.e. they are supposed to process in an interleaved way inputs produced by the user and outputs to be perceived by the user. This explains why the first approaches in formal modelling such systems tried to catch interactive systems dialog by describing the inner states of the systems and how events can affect inner states. Parnas [32] was one of the first to use formal methods based on finite state machines (FSMs) for the specification of human computer interaction issues.

State based formalisms provide a description of all of the states a system can be in and the transitions between these available states. By doing so, such formalisms allow to take into account the interaction context and are therefore more adequate for modelling interactive applications where user actions play a central role.

When WIMP and direct manipulation interfaces appeared the users have been able to carry out several flows of control in a concurrent manner. Formal description techniques used for describing such interaction technique require the explicit handling of parallelism or at least concurrency. Statecharts [16] is such a description technique and have been extended to be able to deal with web interfaces [35]. Petri nets, on which our approach is based, have already been used for modelling interactive systems [5], [22]. They are particularly relevant to describe multimodal interactive systems as they have a true concurrency semantics that is more adequate than an interleaving one especially when independent and parallel control flows are described.

2.2 Modelling Multimodal Interaction and Multimodal Interfaces

Multimodal interactive systems modelling has been addressed in several contribution (mainly focusing on handling inputs) using either flow charts [34] and [12] or Petri nets [18]. In that paper Petri nets are used to describe two handed interaction with a stylus and a Wacom tablet. Further works present approaches able to describe multimodal interaction with widgets or post-WIMP applications, such as Shadow [19] that exploits hierarchical state transition diagram. Modelling multimodality is still maturing but in more recent proposed approaches, formal method has been applied to safety critical interactive systems, as for instance in [3] where the description of a multimodal interactive cockpit of a military aircraft is presented.

2.3 Modelling Fusion Engines

Fusion is a particular component of a multimodal interactive system that typically aims at synchronizing several flows of information produced by the user while interacting with the input devices. For instance, in Johnston and Bangalore [20] both each modality and the fusion engine itself are modelled by automata. This synchronisation is sometimes depending on the elapsed time between the use of two modalities (usually called temporal window). In such a case the description technique must be able to represent quantitative time information This is what has been proposed by Latoschik in [24] who extends the Johnston and Bangalore work with tATN (temporal augmented transition network) for speech and gesture fusion in Virtual reality environment. More information about fusion engine modelling is available in [23] where the set of characteristics are presented and previous work in the field is positioned with respect to them.

3. MODELLING FUSION ENGINES WITH ICOS

As stated above, a fusion engine is a software component in charge of fusing information produced by users when interacting with input devices. That software component and its positioning with respect to interactive systems architecture is presented in this section. We also present how the ICO formal description technique can be used for modelling fusion engines and more precisely we present useful basic constructs expressed in ICO. We then present examples of fusion modelling and lastly discuss fusion politics and how they can be expressed with ICOs.

3.1 Where are fusion engines?

The Arch model [2] was designed to overcome several problems raised by the Seeheim model [15] that defines the three basic components of interactive applications: functional core, presentation and dialog controller.

Arch refines the relationship between these three components and introduces two additional ones:

- The **Physical interaction** aims at handling interaction at a lexical level. It forwards the inputs from users on interactive objects (by acting on the input devices) and performs the rendering of information provided by the logical interaction component. This component is typically addressed by graphical toolkits, while the **Logical interaction** converts information provided by the physical interaction component into abstract (interface independent) information which is then transmitted to the dialog controller (and back again).
- The **Dialog controller** ensures the sequencing of the set of events. It describes the set of the authorized events, defines (according to the inner state of the interactive application represented in the dialog controller too) how the events change

the inner state of the application and how the application reacts to these events.

- The **Functional core adapter** translates the calls from the dialog controller into functional core compliant instructions, and back again.

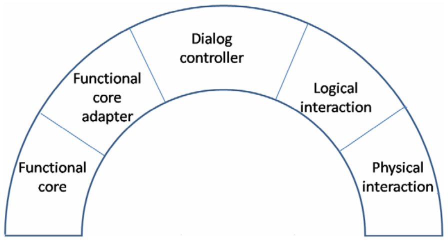

Figure 1. Arch Model.

The event flow from the user actions to the dialog of the application may be the result of transformation or fusion of events across the two following levels at a different levels of abstraction from physical-level events (directly produced by input devices) to dialog-level (or task-level) events:

- The Physical level of events corresponds to events used as input of the physical interaction part of the Arch architecture. It creates **raw input events** from reading the input devices state. By combining and refining these raw events, **higher level events** such as clicks, drag are produced, and by introducing graphical objects these events reach a much higher level. The very simple example of a classical push button illustrates this refinement of the event flow: the related event (for instance *actionPerformed* within Java) is produced after the occurrence of a mouse clicked over the rectangle of the button or pressing the space bar from the keyboard while the button is focused. For instance, in [26] we fused raw data from a Data glove and a motion capture sensor to produce mouse-level events.
- The Dialog level of events corresponds to the management of the connection between the highest level events and domain objects. For instance, in [20] the authors use speech command with the selection of a set of persons using a touch screen to send them an email.

As stated in the related work section, fusion engines may cover different level of this architecture. Section 3.4 presents how models are positioned with respect to Arch depending on the event they produce and the ones they consume.

3.2 ICO: A Petri nets-based Notation

ICOs (Interactive Cooperatives Objects) are a formal description technique dedicated to the specification of interactive systems. It uses concepts borrowed from the object-oriented approach (dynamic instantiation, classification, encapsulation, inheritance, client/server relationship) to describe the structural or static aspects of systems, and uses high-level Petri nets [14] to describe their dynamics or behaviour. The ICO notation is based on a behavioural description of the interactive system using the Cooperative objects formalism that describes how the object reacts to external stimuli according to its inner state. This behaviour, called the Object Control Structure (ObCS) is described by means of Object Petri Net (OPN). An ObCS can have multiple places and transitions that are linked with arcs as with standard Petri nets. As an extension to these standard arcs, ICO allows using *test* arcs and *inhibitor* arcs. Each place has an initial marking (represented by one or several tokens in the place) describing the initial state of the system.

As an extension to standard Petri nets, the Cooperative Objects provide mechanisms for instantiation. Indeed, every ObCS can be instantiated and allows multiple executions of the same class as this is the case in object-oriented approaches. These instantiations can be parameterised by constructor arguments that associate a marking to the instantiation. For example, in an interaction technique exploiting several mice (such interaction may be found in interactive cockpits such as the Airbus A380), each mouse driver is a distinct instance of an ObCS class with different Class Parameters (i.e. the number of the mouse). This makes it possible for each driver to handle its own coordinates represented in the values of the tokens defining the marking of the instance.

Another important part of an ICO description is the link with the presentation part of the interactive system. This is done by means of a tabular description defining two functions, namely Rendering and Activation functions. The Rendering function describes how state changes (represented by tokens movements across the ObCS) trigger presentation methods call (which can be graphical, audio, ..). The Activation function describes how availability of particular transitions leads to availability of user events on the interface. As the paper mainly focuses on behavioural aspects, we do not describe them further (more can be found in [3], [26].

It is important to note that ICOs have been used for other types of interfaces than multimodal ones. The notation is supported by a CASE tool called PetShop [4]. How the tool is structured and how it is integrated in a software development (and in particular with prototyping activities) goes beyond the scope of this paper that focuses on the fusion engines aspects. More information about that aspect is available in [31].

3.3 ICO description of fusion engines bricks

Behavioural modelling of fusion engines requires specific constructs the most salient being sequencing, alternative, parallelism and quantitative time representation.

Petri nets model a discrete event system using state variables, called places (depicted as ellipses) and state changing operators called transitions (depicted as rectangles). The state of the system is modelled by the distribution of tokens in the places of the net (these tokens being able to hold values). Arcs represent the necessary pre and post conditions for a state change to occur as well as how the state changes are performed. The dynamic behaviour of a marked Petri net is expressed by means of two rules, the enabling rule and the firing rule. A transition is enabled (also called fireable) if each input place (places linked to the transition by input arcs) contains at least as many tokens as the weight of the input arcs it is linked to. When an enabled transition is fired, tokens are removed from the input places of the transition and new tokens are deposited into the output places (the number of tokens is defined by the weight on the output arcs).

These rules illustrate one of the main properties of Petri nets, called locality of enabling and firing of transitions, which makes it possible to Petri nets to model true concurrent systems as the firing of a transition only impacts its input and output places leaving the rest of the model unchanged. Thus, if the independent parts of the model are fired by means of parallel processors, the net evolves according to a true concurrency semantics.

The following sections present how it is possible to use the ICO description technique to describe examples of each of the constructs introduced.

3.3.1 Sequencing

Modelling a sequence of events is modelling a qualitative temporal relationship between these events. An example of such sequencing of events occurs when editing a graphical object. The user first selects a graphical object using the dedicated pointing device before triggering the edit command, using for instance a voice command.

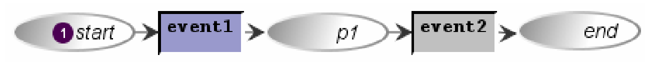

Figure 2. ICO model of a sequence of events.

Figure 2 presents an example of sequence of two events, *event1* and *event2* that behaves as follow: at the beginning, a token is in the place *start* making *event1* the only available event. When *event1* occurs*,* the transition event1 (which is connected to it via the Activitaion function) is fired. During this firing process, the token in place *start* is destroyed and a new token is created and deposited in place *p1*. From that state, the transition event2 becomes fireable and the *event2* is now allowed. When the event *event2* occurs, transition event2 is fired, resulting in the destruction of the token in place *p1* and in the creation of a new token set in place *end*. It is important to note that not all the transitions in an ICO model are related to events. Such transitions are called synchronized transitions (and depicted with at black thick border) and are used to represent even-driven behaviour. Standard transitions are used to model autonomous behaviours. Transition Parallelism_begin in Figure 4 is an example of such an autonomous transition.

3.3.2 Alternative

Modelling alternatives between events is modelling how occurrence of events leads to explicit choices in state changes.

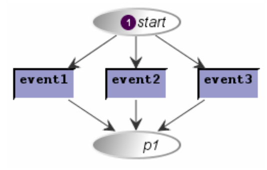

Figure 3. ICO model of alternative.

In Figure 3, the ICO model represents the fact that any occurrence of one of the events *event1, event2* and *event3* will change the system state from having one token in place *start* to having one token in place *p1*. This is useful for letting several options to the user for reaching a desired state in the system. According to CARE properties [11] this would correspond to Equivalent modalities.

3.3.3 Parallelism

Parallelism in a model makes it possible to represent independent behaviours that can occur simultaneously. For instance, an option for describing the classical "put that there" interaction technique could be to allow selecting objects positions while recognizing the speech commands.

An example of parallel handling of events is presented on Figure 4. In this example, *event1* and *event2* may occur independently, but the command corresponding to transition Parallelism_end is only triggered when both events have occurred. When *Parallelism_begin* occurs, a token is set in place *waitfor_event1* and another token is set in place *waitfor_event2*, making available transitions events, *event1* and *event2*. Firing *event1* sets a token in place *event1_received* while firing *event2* sets a token in place *event2_received*. When each of these places contains a token, the transition *Parallelism_end* (representing a synchronisation also called a reduction of parallelism) is fireable and its firing leads to to a state with one token in the place *p1*. According to CARE properties [11] this would correspond to Equivalent modalities.

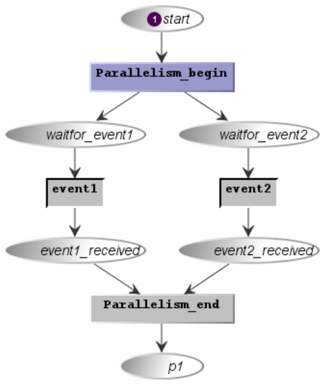

Figure 4. ICO model of parallelism.

3.3.4 Time representation

Fusion engine behaviour usually embeds quantitative temporal evolutions. For instance, one might need to represent the fact that a multimodal command (made up with speech and mouse pointing) is only triggered if the two events occur within a given time frame (for instance 800 milliseconds). One of the advantages of Petri nets with respect to other formal description techniques is their capability to handle in a very efficient way both qualitative temporal aspects (before, after, meanwhile, …) as shown before, and quantitative temporal aspects. In ICO, quantitative time is represented by a time interval attached to a transition (for instance transition timeout in Figure 5) where the time interval can be a constant or a function of the marking of the input place.

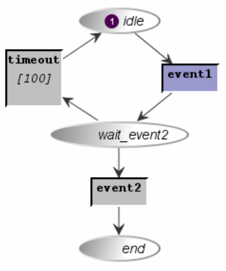

Figure 5. Time Representation.

In the example of Figure 5, after the firing of transition *event1*, a token is set in place *wait_event2*. Two transitions are now fireable, *timeout* and *event2*. *Timeout* is a temporised transition with a time interval of 100 milliseconds (value displayed between square brackets), and if *event2* is not received before these 100 milliseconds, the timeout transition is fired, removing the token from place *wait_event2* and setting a token in place *idle*. If *event2* is received before 100 milliseconds, *event2* is fired and a token is set in place *end* (the token in place *wait_event2* is destroyed too). Such construct is useful in the field of multimodal interfaces to represent alternative behaviours in the fusion engine (for instance if some events are missing for triggering commands). It is also useful for fine-tuning of the fusion engine making it possible to adapt the behaviour to the preferences or the capabilities of the users.

3.4 Modelling Fusion using ICO

ICOs have been used for the formal description of many interaction techniques including gesture, voice, … and also for multimodal interactions techniques such as two-handed, voice and speech … According to the multimodal interaction technique under consideration, fusion of information from the input devices can be performed at different levels of the architecture presented in section 3.1. The following sections present two possible ways of fusing the information using ICOs at different levels. The models correspond to a Virtual chess case study (that has been originally introduced in [26]). In that example, a user may interact with chess pieces using a data glove for identifying finger gestures and a flock of bird (a motion sensor) to locate the hand in a 3D environment.

3.4.1 Fusion at Physical Interaction level

Behaviours at the Physical Interaction level of the architecture deal directly with input devices information flow and input devices states. That data is then processed for building and raising events that are then captured and processed by models embedded in components located higher in the architecture.

Figure 6 illustrates how it is possible to model (using ICO) such event production. That model reads raw data from the input devices (the motion sensor and the data glove) and then builds and raises three logical-level events: *pick(p), move(p)* and *drop(p)*, where *p* represents a point within the 3D environment. The coordinates of that point are determined using data from the motion sensor. User's gesture is captured using the data glove and this is exploited to detect whether a pick or a drop event have to be produced.

The interesting aspect of the flock of bird and of the data glove input devices is that they do not produce events. Instead, they store information about the last gestures in data slots. In order to determine which gestures have been performed by the user, the fusion engine model has to get that data from them. This is performed on a regular basis using a self-triggered event called *idle* (the event *idle* is produced by the internal loop implemented by most graphic libraries, such as OpenGL). It would have been possible to build another model requesting that data at regular time intervals using time-based evolution in ICOs as presented in 3.3.4.

When the event Idle occurs, the transition *idle* is fired and captures the current fingers' flexion from the data glove and the spatial hand's position from the motion captor (this is represented by the method calls fobGetPosition(FOB) and fdGetGesture(FD)). The information concerning the flexion of the fingers and the position of the hand are stored in variables *g* and *p*, respectively. The transitions *pick, notPick, drop* and *notDrop* compare the current and previous positions (which are stored in the token from the place *last*). If the current position is different from the previous one, and the hand is opened, the system triggers an event *drop(p)*. If the hand is closed and its position is different from the previous one, then the system triggers an event *move(p)*. It is important to note that the model in Figure 6 only represents the fusion engine. Indeed, at the Physical Interaction level, the domain objects (such as the chess pieces in our example) are not available and thus the application methods cannot be invoked. Another model (located in the dialog component) should be built to make explicit the connection between the position of the FOB (embedded in the events produced via the variable p) and the pieces on the chessboard. As this paper focuses on the modelling of fusion engines, it is not presented here.

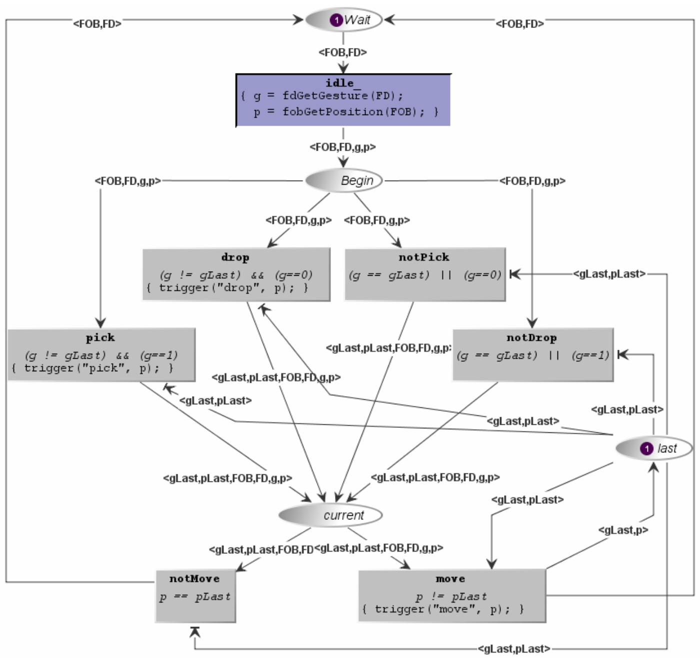

Figure 6. Production of events from low-level device information

3.4.2 Fusion at Dialog Level

According to the Arch architecture, the Dialog component must remain independent from the Physical Interaction component (i.e. input devices in our case). We address this in the models by splitting the fusion engine in two different parts. Communication between these two parts is event-based i.e. the events are produced by the low-level model and caught by the upper-level one. For sake of separation of concerns, we present the models handling the two input devices completely separated. We could have integrated them and exploited parallelism mechanism (presented in section 3.3.3) offered by ICO and the model would have been behaviourally equivalent to the two models presented here.

In the current section, we propose to have one model for each input device and an additional model in charge of fusing information produced by the input device models. The fusion engine is located at the Dialog level as it is able to access domain information such as the location of the pieces on the chessboard. It is important to note that such information is not related to the user interface but to the application domain. Indeed, in a chess game there is always a chessboard with pieces. How that information is displayed and how the user can interact with them belong to the user interface part.

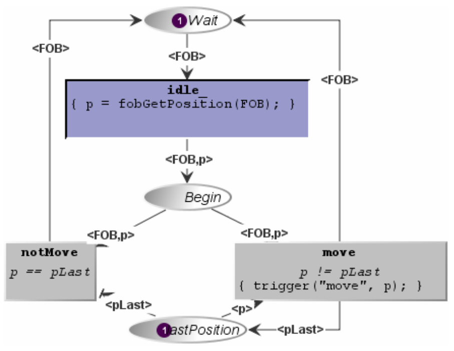

Figure 7. Production of move event from motion sensor data

The behaviour of the event production is very close to the one in the previous section. The logical event is produced by polling the motion sensor's state. Every time an event *idle* is triggered, it fires the transition *idle* to capture the hand's position (Figure 7). The position of the hand (that was stored by the input device) is stored in variable *p*. If and only if the current position is different from the previous one, the model triggers an event *move(p)* (this is represented in the model by the preconditions (p==plast and p!=pLast) in transitions move and notMove.

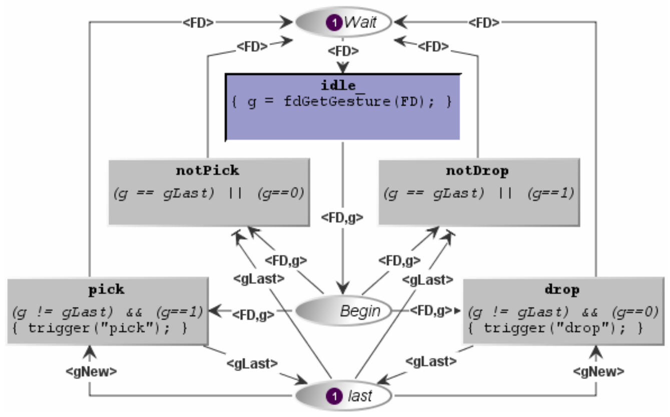

Figure 8. Production of pick and drop events from gesture detection

In the same way, for Figure 8, every time an event *idle* is triggered the current fingers' flexion from the data glove is captured and its value is stored in variable *g*. The transitions *pick, notPick, drop* and *notDrop* compare the current and previous gesture (which is given by the token from the place *last*). For instance, the transition drop describes the fact that if the current gesture is different from the previous one (g!=gLast), and the hand is opened (g==0) then the system triggers an event *drop* (trigger("drop")).

Fusion is performed as modelled in Figure 9 where the two event flows are handled separately, the move(p) event on one side and the pick and drop events on the other side.

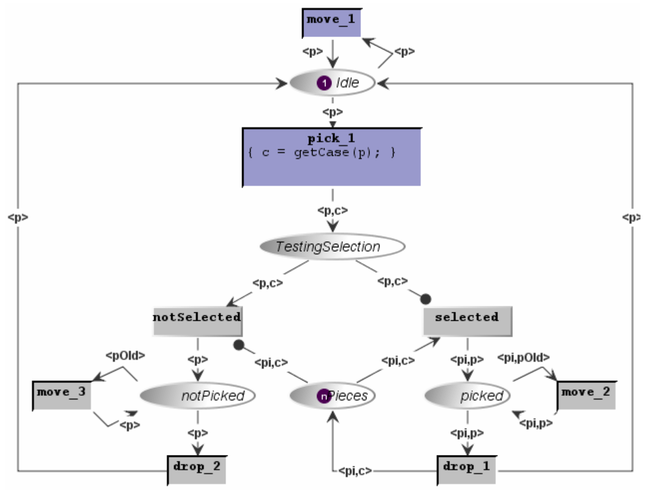

Figure 9. Handling the event flows in the fusion engine

Independent from the input device used (data glove or motion captor), the dialogue controller receives produced events (*pick, drop* and *move(p)*). As represented in Figure 9, when receiving event move (one of the transition move_1, move_2 or move_3 will be available) the current coordinates are updated (in place Idle, notPicked or Picked depending on the transition fired). After the reception of move, when an event *pick* occurs (transition *pick_1*) the fusion of the two events is possible (i.e. both the position and the related command pick are known). In order to perform the fusion, the fusion engines need to know whether or not there is a piece on the chessboard at position p. This is performed in transition pick_1 (code c=getCase(p)).

The variable c is used to store the value of the piece at position p. From that state two options are possible: either c contains a reference to a piece or c is empty. First case is handled by the right-hand side of Figure 9 while the other one is handled by the left-hand side. If an event *pick* occurs and the place *pieces* contains a reference to square *c*, then the user can move the corresponding piece *p* (using the transition *move_2*) or drop it (using the transition *drop_1)*. Otherwise, the user can only move the hand over the chessboard for a while (transition move_3) and then the system return to the initial state. This modelling exploits the alternative presented in section 3.3.2.

3.5 Ambiguity Resolution in Models

The example presented above has shown that using ICOs it is possible to perform fusion at different levels (according to the software architecture Arch). This section extends previous modelling capabilities by providing several examples of models making explicit how the ambiguity resolution can be modelled. When modelling a complete fusion engine, the construction of a command may require solving ambiguities caused by the occurrence or non-occurrence of events or by the fact that event might not hold consist of values. For instance, in a "Put that there" interaction (using speech and mouse) production of the command requires both the occurrence of the speech command ("put") and the two positions (the first one corresponding to the position of an object and the second one corresponding to the position of position on the screen). Modelling fusion engines require the handling of potentially incomplete or inconsistent multimodal events.

Even if policies of fusion are out of the scope of this paper, we present in the following sections how it is possible to ICO for modelling ambiguities resolution.

3.5.1 Temporal windows

The use of temporal windows is one of the most used policies for ambiguity resolution in the literature. The basic principle is to allow the fusion of several events only if these events occur within a given temporal window. The temporal window proposed in [28] is modelled in Figure 10.

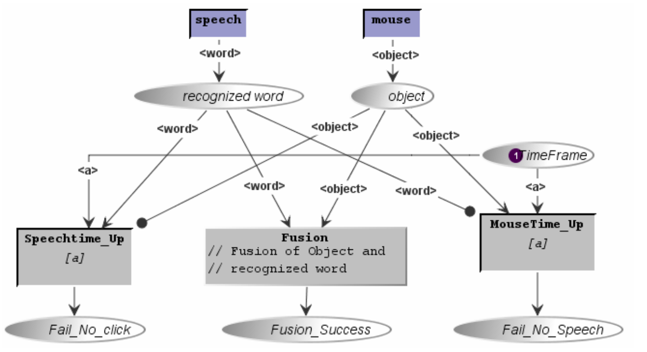

Figure 10 Fusion model based on a temporal window

In Figure 10, when a speech command is received, a token is set in place *recognized word* and when a mouse command is received a token is set in place *object*. If both places hold a token before the timeout (modelled by the value of the token in place *Timeframe*), the transition *fusion* is fired performing the fusion (not detailed on the model). On the other side, if a token is set in the place *object* (resp. place *recognized word*) while no token is set in the place *recognized word* (resp. place *object*) within timeout, the timed transition *MouseTime_Up* (resp. *SpeechTime_Up*) removes the token from the place *object* and place it in *Fail_No_Speech* (resp. in the *Fail_No_Click*).

3.5.2 Unification with a set of rules

Another classical policy is the unification of several events according to a set of rules as presented in [10]. Figure 11 illustrates the modelling of such unification using the same event sources as in the previous section.

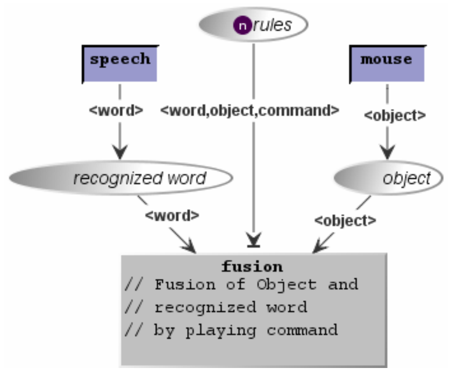

Figure 11 Fusion model based on unification with rules

When a speech command is received, a token is set in place *recognized word* and when a mouse command is received a token is set in place *object*. If both places hold a token, the transition *fusion* can be fired. The actual firing depends on the values of the tokens. If the pair composed by the word recognized (represented by the variable *word* on the arc from place *recognized_word* to transition *fusion*) and the object linked to the mouse event (represented by the variable *object* on the arc from place *object* to transition *fusion*) can be unified with one of the tokens hold in place *rules* then the firing can occur. In such a case, the corresponding fusion can be performed according to the value of the object and the one of the word recognized, applying the command hold by the token form place *rules*, identified by unification. In such models, the labels on the arcs of the net are used as a mean of changing the behaviour of the model not only based on the presence of tokens but also on the values carried by the tokens. The rules are expressed by means of triplets composed of a word, an object and a command. An instance could be ("remove", rectangle, delete). The command delete is triggered when both words remove and rectangle are uttered.

3.5.3 Modelling complex fusion mechanisms

The previous examples illustrate basic policies applicable to fusion engines. Modelling a complex fusion engine obviously requires to compose such policies. For instance, the example of Figure 11 can be easily extended by introducing the timeout part of the temporal window of Figure 10 allowing the description of unification together with a quantitative temporal evolution. Using ICO allows the description of complex fusion engines, for instance, it is possible to describe the behaviour of fusion by agent, adding intelligence in the fusion engine by, for instance, improving the content of the command hold by tokens in place *rules* on Figure 11.

4. PERSPECTIVES AND RESEARCH AGENDA

Describing fusion engines using a formal description technique makes it possible to verify its behaviour according to properties that might have identified as pertinent. Such properties can be generic to any fusion engine (every event received is processed) or specific to a given design context. Some formal description techniques are available for describing such properties but their verification remains a complex task with little support from tools.

Another benefit of using formal description technique is that it is possible to assess the performance of the models. Here again performance evaluation techniques are available and more work has to be performed before reaching the adequate tool support for the assessment of the performance of fusion engines in an industrial context.

Lastly, the fusion engine is a critical element of the multimodal interface and, as such, must be assessed with respect to the usability of the resulting interface. Such assessment can be supported by formal description techniques by (for instance) making it possible to generate the usage scenarios [7] and to identify the ones requiring to be tested using usability evaluation techniques [6].

5. CONCLUSION

This paper has presented how fusion engines can be modelled using ICOs which is a Petri net based formal description technique dedicated to the modelling, verification and simulation of interactive systems. The paper has emphasised how some of the constructs of the formal description technique fit with the needs for fusion engine modelling. The examples given have presented in detail how fusion engines modelled with ICOs can exhibit behaviours including parallelism, unification, quantitative and qualitative temporal evolution, …

This work belong to more ambitious research programme aiming at producing methods, tools and techniques for the engineering of multimodal interfaces in the field of safety critical interactive systems. Indeed, ICOs provide a complete, concise and un-ambiguous description of the fusion engine that makes it possible to assess the performance, the efficiency and the reliability of multimodal interfaces thus providing a way of broadening the application of multimodal interfaces.

6. ACKNOWLEDGMENTS

This research is partly funded by the CNES R&T Tortuga project, R-S08/BS-0003-029 and the Network of Excellence ReSIST IST-4-026764-NoE (www.resist-noe.org).

7. REFERENCES

[1] Barboni E., Conversy S., Navarre D., Palanque P. Model-Based Engineering of Widgets, User Applications and Servers Compliant with ARINC 661 Specification. Design Specification and Verification of Interactive Systems (DSV-IS 2006), LNCS, pp. 25-38.

[2] Bass, L., Pellegrino, R., Reed, S., Seacord, R., Sheppard, R., and Szezur, M. R. The Arch model: Seeheim revisited. proceeding of the User Interface Developpers' workshop. 91.

[3] Bastide R., Navarre D., Palanque P., Schyn A., Dragicevic P. A Model-Based Approach for Real-Time Embedded Multimodal Systems in

Military Aircrafts. In : ICMI 2004 - Sixth International Conference on Multimodal Interfaces, ACM Press, pp. 243-250.

[4] Bastide, R., Navarre, D., and Palanque, P. 2002. A model-based tool for interactive prototyping of highly interactive applications. In CHI '02 Extended Abstracts on Human Factors in Computing Systems (Minneapolis, Minnesota, USA, April 20 - 25, 2002). CHI '02. ACM, New York, NY, 516-517

[5] Bastide R. & Palanque P. Petri nets with objects for specification, design and validation of user-driven interfaces. In proceedings of the third IFIP conference on Human-Computer Interaction, Interact'90. Cambridge 27-31 August 1990.

[6] Bernhaupt R., Palanque P., Winckler M., Navarre D. 2007. Usability Study of Multi-Modal Interfaces using Eye-Tracking. Proceedings of INTERACT 2007. LNCS 6643, Springer Verlag.

[7] Bernhaupt R., Navarre D., Palanque P., Winckler M. 2007. Model-Based Evaluation: A New Way to Support Usability Evaluation of Multimodal Interactive Applications. In “Maturing Usability: Quality in Software, Interaction and Quality” Springer Verlag, April 2007.

[8] Bolt R. A., “Put-that-there”: Voice and gesture at the graphics interface, ACM SIGGRAPH Computer Graphics, v.14 n.3, p.262-270, July 1980

[9] Bouchet, J., Nigay, L., and Ganille, T. 2004. ICARE software components for rapidly developing multimodal interfaces. In Proceedings of the 6th international Conference on Multimodal interfaces (State College, PA, USA, October 13 - 15, 2004). ICMI '04. ACM, New York, NY, 251-258.

[10] Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. 1997. QuickSet: multimodal interaction for distributed applications. In *Proceedings of the Fifth ACM international Conference on Multimedia*. MULTIMEDIA '97. ACM, New York, NY, 31-40.

[11] Coutaz J., Nigay L., Salber D., Blandford A., May J., Young R. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties. In Proceedings of the INTERACT'95 conference. pages 115-120. 1995. S. A. Arnesen & D. Gilmore Eds., Chapman & Hall Publ., Lillehammer, Norway.

[12] Esteban, O., Chatty, S. & Palanque P, 1995, Whizz'Ed: a visual environment for building highly interactive interfaces. Proceedings of the Interact'95 conference. 1995, p121-126.

[13] Flippo, F., Krebs, A., and Marsic, I. 2003. A framework for rapid development of multimodal interfaces. In *Proceedings of the 5th international Conference on Multimodal interfaces* (Vancouver, British Columbia, Canada, November 05 - 07, 2003). ICMI '03. ACM, New York, NY, 109-116.

[14] Genrich, H. J. 1991. Predicate/Transitions Nets. In High-Levels Petri Nets: Theory and Application . K. Jensen and G. Rozenberg (Eds.)Berlin: Springer Verlag (1991) pp. 3-43.

[15] Green, M., "A survey of three dialogue models." 1986. ACM Trans. Graph., n°3, Vol. 5, pp. 244□□275.

[16] Harel, D. 1987, "Statecharts: A visual formalism for complex systems." Sci. Comput. Program., Elsevier North□Holland, Inc., 1987, n°3, Vol. 8, pp. 231□274.

[17] Heymann, Michael and Degani, Asaf., "Formal Analysis and Automatic Generation of User Interfaces: Approach, Methodology, and an Algorithm." Human Factors: The Journal of the Human Factors and Ergonomics Society, Vol. 49, pp. 311□330.

[18] Hinckley, K., Czerwinski, M. and Sinclair, M., 1998. Interaction and modeling techniques for desktop two-handed input. In Proceedings of the 11th Annual ACM Symposium on User interface Software and Technology. UIST '98. ACM, New York, NY, 49-58.

[19] Jacob, R. J., Deligiannidis, L., and Morrison, S. 1999. A software model and specification language for non-WIMP user interfaces. ACM Trans. Comput.-Hum. Interact. 6, 1 (Mar. 1999), 1-46.

[20] Johnston, M. and Bangalore, S. 2005. Finite-state multimodal integration and understanding. Nat. Lang. Eng. 11, 2, 159-187

[21] Krahnstoever, N., Kettebekov, S., Yeasin, M., and Sharma, R. 2002. A Real-Time Framework for Natural Multimodal Interaction with Large Screen Displays. In Proceedings of the 4th IEEE international Conference on Multimodal interfaces - Volume 00 (October 14 - 16, 2002). International Conference on Multimodal Interfaces. IEEE Computer Society, Washington, DC, 349.

[22] Keh, H. C. and Lewis, T. G., 1991. Direct-manipulation user interface modeling with high-level Petri nets. In Proceedings of the 19th Annual Conference on Computer Science (San Antonio, Texas, United States). CSC '91. ACM, New York, NY, 487-495.

[23] Lalanne D., Nigay L., Palanque P., Robinson P., Vanderdonckt J., Ladry J-F 2009 Fusion Engines for Multimodal Interfaces: a survey. International Conference on Multimodal Interfaces and Workshop on Machine Learning for Multi-modal Interaction (ICMI-MLMI 2009), Cambridge, Massachusetts, USA, 2009, ACM, *this volume*.

[24] Latoschik, M.E., 2002. Designing transition networks for multimodal VR-interactions using a markup language. Multimodal Interfaces, 2002. In Proceedings of the Fourth IEEE International Conference. pp. 411-416, 2002.

[25] Mellor S. and Balcer M. 2002 Executable UML: A Foundation for Model-Driven Architectures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[26] Navarre D., Palanque P., Bastide R., Schyn A., Winckler M., Nedel L., Freitas C. 2005 A Formal Description of Multimodal Interaction Techniques for Immersive Virtual Reality Applications. In : Human-Computer Interaction - INTERACT 2005: IFIP TC13 International Conference, Springer-Verlag, pp. 170-185, September 2005.

[27] Neal, J. G., Thielman, C. Y., Dobes, Z., Haller, S. M., and Shapiro, S. C. Natural language with integrated deictic and graphic gestures. In Proceedings of the 1989 Workshop on Speech and Natural Language. Human Language Technology Conference. Association for Computational Linguistics, Morristown, NJ, 410-423

[28] Nigay, L. and Coutaz, J. 1995. A generic platform for addressing the multimodal challenge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI’95). ACM Press/Addison-Wesley Publishing Co., New York, NY, 98-105.

[29] Object Management Group: Unified Modelling Language (UML) 2.0 Superstructure Specification, August 2003. Ptc/03-08-02, pp. 455-510

[30] Oviatt, S. Ten myths of Multimodal Interaction Communication of the ACM; 42: 11: 74-81, 1999.

[31] Palanque P., Ladry J-F, Navarre D. & Barboni E. High-Fidelity Prototyping of Interactive Systems can be Formal too 13th Int. Conf. on Human-Computer Interaction (HCI International 2009) San Diego, CA, USA, Springer Verlag, LNCS 5610.

[32] Parnas, D. L., 1969. On the use of transition diagrams in the design of a user interface for an interactive computer system. In Proceedings 24th National ACM Conference. pp. 379-385. 1969

[33] Sun, Y., Chen, F., Shi, Y., and Chung, V. 2006. A novel method for multi-sensory data fusion in multimodal human computer interaction. In Proceedings of the 18th Australia Conference on Computer-Human interaction. OZCHI '06, vol. 206. ACM, 401-404.

[34] Willans, J. S. and Harrison, M. D., 2001. Prototyping Pre-implementation Designs of Virtual Environment Behaviour. In Proceedings of the 8th IFIP international Conference on Engineering For Human- Computer interaction (May 11 - 13, 2001). LNCS, vol. 2254. Springer-Verlag, 91-108.

[35] Winckler, M.; Palanque, P. 2003. StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications. International Workshop on Design, Specification and Verification of Interactive Systems - (DSVIS'2003), LNCS n° 2669