Formal description of multi-touch interactions
Arnaud Hamon, Philippe Palanque, José Luís Silva, Yannick Deleris, Eric Barboni

To cite this version:

HAL Id: hal-04083399
https://hal.science/hal-04083399
Submitted on 27 Apr 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 12677

To link to this article: DOI: 10.1145/2494603.2480311
URL: http://dx.doi.org/10.1145/2494603.2480311


Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr
Formal Description of Multi-Touch Interactions

Arnaud Hamon\textsuperscript{1,2}, Philippe Palanque\textsuperscript{2}, José Luís Silva\textsuperscript{2}, Yannick Deleris\textsuperscript{1}, Eric Barboni\textsuperscript{2}

\textsuperscript{1}AIRBUS Operations, 316 Route de Bayonne, 31060, Toulouse, France
\textsuperscript{2}ICS-IRIT, University of Toulouse, 118 Route de Narbonne, F-31062, Toulouse, France

(hamon, palanque, silva, barboni)@irit.fr, yannick.deleris@airbus.com

ABSTRACT

The widespread use of multi-touch devices and the large amount of research that has been carried out around them has made this technology mature in a very short amount of time. This makes it possible to consider multi-touch interactions in the context of safety critical systems. Indeed, beyond this technical aspect, multi-touch interactions present significant benefits such as input-output integration, reduction of physical space, sophisticated multi-modal interaction … However, interactive cockpits belonging to the class of safety critical systems, development processes and methods used in the mass market industry are not suitable as they usually focus on usability and user experience factors upstaging dependability. This paper presents a tool-supported model-based approach suitable for the development of interactive systems featuring multi-touch interactions techniques. We demonstrate the possibility to describe touch interaction techniques in a complete and unambiguous way and that the formal description technique is amenable to verification. The capabilities of the notation is demonstrated over two different interaction techniques (namely Pitch and Tap and Hold) together with a software architecture explaining how these interaction techniques can be embedded in an interactive application.

Author keywords

Tactile interactions, development process, model-based approaches, interactive cockpits

INTRODUCTION

The industrial and academic world have been providing prototypes, toolkits and toy systems offering tactile and more recently multi-touch interaction techniques for more than two decades now. However, the actual engineering of multi-touch 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 interactive systems featuring multi-touch interactions.

While identifying requirements and user needs for user interfaces in the area of command and control for safety critical systems the designers have to decide either to go for systems with standard and (usually) poor interaction techniques or to embed new (and more sophisticated) interaction techniques. If the users’ tasks are complex, requiring, for instance, the execution of multiple commands in a short period of time or the manipulation of large data sets, it is likely that the new interaction techniques will significantly improve the overall performance of the operators. However, in such cases, the development process will at the minimum be more difficult (resources consumption will increase throughout the design, development and evaluation stages) or even be impossible if tools and techniques available for the development do not bring the required level of quality in the final product. In the case of safety critical systems, quality is assessed by the dependability level of the interactive system which must be compliant with the requirements set by the certification authorities.

Beyond the fact that they have reached the adequate maturity level, multi-touch interaction techniques present a set of advantages as identified in [12]:

- The screen content can be completely modified in order to include input management previously devoted to hardware input devices such as keyboard or mouse
- They are by nature multimodal systems taking advantage of these interaction techniques These previous studies (and additional ones such as [14] and [37]) 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 as it decreases 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.

- It is possible to embed a lot of detailed information within a single input such as pressure, orientation of the finger (using the shape of the fingertip) [29];
- They offer a very easy forum for multi-user interaction reducing articulatory coordination effort that is required if input devices are to be shared.

![Figure 1 - High level representation of a cockpit](image)

As visible on the Figure 1, the cockpit is made up of 6 large display units, 2 head-up displays and 2 Keyboard Cursor Control Units (an input device integrating a keyboard and a track ball). This paper is part of a study assessing the possible implementation of a map application currently available in the On Board Information System (IOS) with multi-touch interactions. This change of interaction technique (from a standard WIMP interaction as promoted by ARINC 661 specification [4]) to the Multi-Function Display (MFD) which is located in-between the captain and the first officer seats thus allowing collaborative tasks between the two pilots on this shared workspace. However, in order to deploy such interaction techniques in the cockpit of commercial aircraft, it is required to ensure that the dependability level of the cockpit is as reliable as the previous cockpits.

Next section presents and compares previous contributions in the field of multi-touch interactions with a special highlight on expressive power of the notations. The ability of the notation to provide verification techniques and to demonstrate properties on the interaction techniques is also exhibited. The following section presents a quick overview of the ICO formal description technique and highlights how this description technique is able cover the needs that have been highlighted in related work section. A multi-levels approach is then presented which is able to transform low-level events produced by the multi-touch device into meaningful events such as *Pinch* or *Tap Long* to be received and handled by the interactive application. Section 4 briefly highlights how properties verification can be addressed. Last section identifies a research agenda for future work that still has to be carried before deploying multi-touch interaction in the cockpits of large civil aircrafts.

**STATE OF THE ART**

In the following paragraphs, we first detail the different conceptual decompositions of multi-touch interactions taking a linguistic point of view on multimodal interactions. Then we compare various notations that have been proposed to describe this interaction paradigm. As the main objective of this paper is to provide a notation for engineering dependable multi-touch interactions, we compare several software architectures that have been proposed for enabling the use of such interaction techniques. This related work analysis is then put into perspective using the more generic point of view of multimodality concepts.

**Multi-touch interactions as a language**

Linguistic point of view, such as semiotics (description of all phenomena associated with the production and interpretation of signs and symbols) are used in [25] to describe multi-touch gestures. However, this semiotics approach only encompasses some of the multi-touch features not addressing explicitly the production of higher-level events (such as double taps) from low-level events (touch, move, up). This is why, compared to [25], we are following a standard linguistic view based on lexicon, syntax and semantics for addressing multi-touch interactions. The lexicon is composed of the low level users’ events while the syntax describes their combination (potentially fusion in the case of multimodal interactions). Regarding semantics (meaning of the interactions) and pragmatics (user mental model), the present work is based on the same definition exposed in [25]. This decomposition allows sorting our various contributions in this paper as follows:

- Lexicon: various event types - elementary vocabulary of the interaction;
- Syntax: combination of interaction models and fusion model;
- Semantic: the dynamic mapping between interaction technique and system command;
- Pragmatics is beyond the scope of the paper.

**Notations for multi-touch interaction description**

**Description**

Table 1 is an extension of the work presented in [35] with additional properties (Analysis…) and references (CPN…).
It summarizes the expressiveness of the UIDL (User Interface Description Language) through ten different properties of the language that are used to characterize this expressiveness. This expressive power is not a goal per se but it clearly defines the type of user interface that can be described using the UIDL and the ones that are beyond their reach. This paper also adds modularity and formal analysis features as the considered context relies on both usability and reliability aspects of multi-touch. The first three characteristics deal with description of objects and values in the language (this is named “Data Description”), with the description of states (“State Representation”) and the description of events (“Event Representation”). For all characteristics, there are four possible values.

- **Yes** means that that characteristic is explicitly handled by the UIDL;
- **No** means that the characteristic is not explicitly handled;
- **Some** means that the property is partly present; and
- **Code** means that the characteristic is made explicit but only at the code level and is thus not a construct of the UIDL.

For instance, data is described in many UIDLs such as ICON [16], which allows modeling data emission and reception from an output port of a device to the input port of another device. Some UIDLs can also represent states of the system, such as ICon [16], which represents the states with nodes in the models. Events are also sometimes explicitly represented as in Wizz’Ed [18] where connections between bricks represent event flows.

Time is also an important characteristic for behavioral description of interactive applications.

**Time**

**Qualitative** time between two consecutive model elements aims at representing ordering of actions such as precedence, succession, and simultaneity. In OSU [27] a transition between two places represents the fact that the activity represented by the second place will only be active after the first one is achieved. **Quantitative** time between two consecutive model elements represents behavioral temporal evolutions related to a given amount of time (usually expressed in milliseconds). This is necessary for the modeling of the temporal windows in a fusion engine for multimodal interfaces, where events from several input devices are fused only if they are produced within a same time frame. In ICO (in this article), timed transitions express such constraints. Finally, quantitative time over nonconsecutive elements was introduced in [38] for multimouse double and fusion double click interactions.

**Concurrent behavior**

Representation of concurrent behavior is necessary when the interactive systems feature multimodal interactions or can be used simultaneously by several users. This can be made explicit in the models like in data flow notations, as in ICon [16] or Whizz’Ed [18] and in all the notations based on Petri nets (last four columns of Table 1). Concurrency representation can also be found in older languages such as Squeak [13], where it is possible to represent parallel execution of processes. This aspect is critical for multitouch interactions due to the concurrent use of multiple fingers and hands.

<table>
<thead>
<tr>
<th>Capability to deal with multi-touch interactions</th>
<th>Explicit</th>
<th>Implicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multimodality: fusion of several modalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reconfiguration of low level events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reconfiguration of interaction technique</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Widgets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concurrent Behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Representation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qualitative between two consecutive model elements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantitative between two consecutive model elements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantitative over non consecutive elements</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 – UIDL expressiveness and handling of multi-touch interactions
Dynamic instantiation
Dynamic instantiation of interactive objects is a characteristic required for the description of interfaces where objects are not available at the creation of the interface as, for instance, in desktop-like interfaces where new icons are created according to user actions. Supporting explicit representation of dynamic instantiation requires the UIDL to be able to explicitly represent an unbounded number of states, as the newly created objects will by definition represent a new state for the system. Most of the time, this characteristic is handled by means of code and remains outside the UIDLs. Only Petri-nets-based UIDLs can represent explicitly such a characteristic, provided they handle high-level data structures, or objects, as is the case for many dialects [31], [8], [23]. In the multi-touch context, new fingers are detected during execution time. Thus, the description language must be able to receive dynamically created objects. In Petri nets this is particularly easy to represent by the creation/destruction of tokens associated to the objects. This way, for instance, for each finger currently touching the multi-touch surface, a corresponding token will be set in a place of the Petri net.

Dynamcity presented is handled at development time i.e. when the system is designed and build. However, dynamicity has also to be addressed at operation time i.e. when the system is currently in use. For instance, to cope with potential hardware failure reconfigurations of the interaction techniques might be required. In [36] we have presented how such dynamic reconfiguration can be modeled and executed. This corresponds to a meta-level representation of interactions which can be dynamically selected at run-time. This is an important aspect to address if multi-touch interactions have to be embedded in safety critical applications. Moreover, in order to ensure the availability of every system states and maintain a high level of usability, the mapping between interaction techniques and commands (such as presented in a static way in [44]) shall be resolved during run-time.

Multimodality
This row refers to the capability of a language to support the fusion of several distinct modalities such as the combination of pen and multi-touch in [19]. Fusion engines have been a focal point of the research in the area of multimodal interactions and they are of prime importance as far as multi-touch interactions are concerned. A survey about the characteristic of fusion engines can be found in [32] and the requirements expressed there are directly applicable to multi-touch interaction.

Analysis
Analysis of the interaction techniques is a critical aspect in order to reduce time and resources spent on user studies and if reliability is considered an important property of the final system. Typically, analysis requires a formal description of the interactions and can be separated into three groups addressed by different types of analysis techniques:

- **validation**, accomplished by interactive simulation (step by step), invariant, structural and reachability/coverability graph analysis;
- **verification**, accomplished by invariant, structural and reachability/coverability graph analysis;
- **performance analysis**, accomplished by simulation.

The results of the analysis aim at detecting errors in the formal description, to validate the existence of required properties and to study the performance of the proposed interaction techniques.

As stated in Table 1 only few approaches for UIDL provide support analysis. Marigold [49] addresses limited validation and verification analysis based on reachability graph analysis. Verification analysis results are based on the verification of properties such as deadlock-free or liveness and the validation by a step by step interactive simulation of the model. By using time in the models (timed colored Petri nets) CPN Tools [23] provides performance analysis support.

Architectures to support multi-touch
Various software architectures for multi-touch applications have been proposed such as in [24] where a taxonomy describes them. Most interactive software architectures are layer-based [17] in order to enrich low level user events into high level events and then interactions techniques. To resolve the computational delays introduced by these architectures and allowing most immediate feedback which is needed by user during direct manipulation, [17] introduced a low-latency subsystem computing the fingers’ trace to be immediately displayed. Most of these architectures address hardware/software integration. We argue that these technical solutions only provide local solutions to the issue of development of multi-touch interaction technique and the key point is to integrate them seamlessly with the description technique. This is the reason why, this paper proposes a more generic architecture model that enables all features listed in Table 1 and is based on the Arch model [7] represented on Figure 2.

![Figure 2 - Arch Model from [7]](image-url)
MODELLING MULTI TOUCH INTERACTION TECHNIQUES WITH THE ICO FORMALISM

ICO: Informal definition
ICO [Interactive Cooperatives Objects] is 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 [20] to describe their dynamics or behavior. The ICO notation is based on a behavioral 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 behavior, 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 the paper mainly focuses on behavioral aspects, we do not describe them further (more can be found in [35]. 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 [9]. As it goes beyond the scope of this paper that focuses on the fusion engines aspects, more information about the tool structure and integration in a software development process is available in [40].

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 [9]. As it goes beyond the scope of this paper that focuses on the fusion engines aspects, more information about the tool structure and integration in a software development process is available in [40].

The first level corresponds to the low level transducer while the second one is composed by the various interaction technique models. Finally, the fusion engine model ensures consistency between the recognized events and is in charge of triggering these events to the dialog part for system command construction. In the following parts, we will use the following graphical hints to ease the reading of the models’ descriptions: places, events, transitions. As there is a Java binding to ICOs and Petshop the detailed elements are given with respect to that binding. Each element of the architecture is presented in details together with its modeling using the ICO notation introduced above.

Low level transducer
The low level transducer is the one model linked to the hardware touch events. An excerpt of this model is presented Figure 4. It parses the features of the received event into a java finger object. The FingerPool place acts as a limiter on the allowed number of distinct fingers input. This transducer packages events, forwards them to models listeners (i.e. higher level events handlers) such as TapAndHold, Pinch… as defined in [1]. Indeed, a “toucheventf_move” or “toucheventf_up” event will only be triggered if the event corresponds to a registered finger.

Managing the event chain from hardware to application

From raw data events to object manipulation and system commands
As we demonstrated in the section introducing multi-touch architecture principles, the architecture we propose here (see Figure 3) can be directly mapped onto the ARCH architecture presented in Figure 2. The 3 circles in the ARCH model are thus explicitly represented on that architecture.

The following paragraph describes the model of the “standard” interaction technique called Pinch and presented in Figure 5. When the interaction transducer is in initial state, all places of the models are free of tokens. The model may receive the low level event “toucheventf_down” handled by the synchronized transition toucheventf_down_1. When this event occurs, a token is set in the place p1. This token comprises a finger object synthetizing the touch information encompassed by the low level event. Another token (empty this time) is added in the place nbFingerModel and enables to toucheventf_up_1 transition, allowing the model to handle “toucheventf_up” events. In this configuration, two low level events may be handled:

- “toucheventf_down”: another “toucheventf_down” received event behaves the same way on the PetriNet. Then if two token are stored in the p1 place, the eagerFusion transition is automatically crossed, grouping both fingers into the same token in place p2.
“toucheventf_up”: as long as the transducer contains information about at least one finger, the event handler toucheventf_up_1 is fireable. Each time such an event is received, a token containing the corresponding finger information is added to temp place, leading to two cases:

- The “toucheventf_up” event corresponds to a finger stored in place p1: the transition endInteraction1 is fired, removing the finger’s related token in p1 and temp as well as one token from nbFingerModel place.
- The “toucheventf_up” event corresponds to a finger stored in place p2: the transition endInteraction2 is fired, subtracting the finger’s related token in p2 and temp; and two tokens from nbFingerModel place since two fingers are composing tokens in place p2.

While waiting in place p2, the transition toucheventf_move_1 is enabled and can handle move events from the low level transducer. When such an event occurs, the transition is fired and updates the corresponding finger’s information. Finally the transition triggers a “pinch” event.

**Figure 5 - Pinch Interaction transducer’s model**

Combining interaction techniques

The ICO notation together with the related architecture (presented in Figure 3) allows the design of complex interaction techniques using the events triggered by models other interaction techniques such as the uniform scale interaction proposed in [29] and combining a “one-touch on the object, together with a two-touch pinch”. Due to the space constraint, we will not present the corresponding model in this paper. However, its principle is identical to the other interaction technique models.

**Interaction Manager**

The interaction manager acts as a supervisor entitled to generate coherent user events from its lower level transducers towards the application. This model may act as a fusion or fission engine depending on the type of rules it implements. Figure 6 details the model of another role of the interaction manager i.e. conflict management between interaction techniques. Indeed, in early design phases, interaction designers specify standalone interaction techniques which might, in the end, be conflicting. Such conflicts can be identified and corrected later on using regular expression analysis as demonstrated in [28]. We argue that this course of action may alter the usability of the initial standalone interactions in order to cope with local and identified conflicts. The interaction manager aims at resolving these local conflicts preserving usability by implementing simple resolution rules. An example of such conflict may occur when two interaction techniques interfere. For instance Pinch could interfere with a TapAndHold interaction if one of the fingers used for the Pinch does not move enough and thus is treated as a TapAndHold even though involved in a Pinch. Such a scenario is part of the interaction specification process we presented in [21] and applied to the interaction techniques fusion engine. To solve this conflict a possible modification is to give priority to Pinch and thus disabling TapAndHold interaction when a Pinch interaction is being recognized.

**Figure 6 – Interaction Manager**

When the interaction manager receives a “Pinch” event, the synchronized transition Pinch_1 forwards the event (trigger PINCH) and puts two tokens in the PinchingFingers place, each token being compose by an int value corresponding to one of the Pinch finger. In parallel, each time a “TapAnhHold” is received, the event is stored in the TapAndHoldId place with the corresponding finger id. Then, the transitions inhibitTapAndHold and ForwardTapAndHold which are in mutual exclusion test if the finger from the received event is involved in a Pinch interaction and process it according to the rule presented above. When a “toucheventf_up” is received, if it impacts on finger involved in a Pinch interaction, the pinchFingerAffected transition subtracts the PinchingFinger token with the same id; otherwise the transition noFingerAffected discards the token.

In addition of the conflict resolution rules, the interaction manager acts by default as an event forwarder towards the
applications. This allows the applications to be registered only to this one model which keep them independent from the various transducers and their architecture. This forwarding role is instantiated by the trigger actions in the various synchronized transitions.

Figure 7 - ICO services for dynamic registration to low level events providers

Handling multi-touch specificities

Dynamic instantiation and management

During multi-touch interactions, fingers are by definition detected dynamically. We argue that the most natural manner to fully specify multi-touch interactions is for UIDL to support dynamic instantiation (creation of inputs devices and GUI components at run time). Indeed, most operating systems handle plug and play devices. Therefore a notation for multi-touch application specification should be able to dynamically detect and manage input devices. Figure 7 presents one ICO service called addLowLevelProvider. This handles a list of LowLevelProviders (stored in place with the same name) and can be added to the ICO transducer presented in Figure 4. It also allows the transducer to listen to "rawToucheventf_down" (moves and ups) fired by the providers it is a listener of. Due to space constraints, the service to remove providers it not presented.

Finger clustering

In purely multi-touch interaction techniques, determining a correct mapping between fingers of the same hand/user is critical as demonstrated in [30]. Therefore we present how our notation is able to formally address this aspect at run-time. The initial Pinch transducer model Figure 5 matches fingers in the order they are pressed. This specific model works for a single user that interacts with one object at the time. In the context of our application domain the presented model will suffer shortcomings when two users will start to interact on a multi-touch surface at the same time. The model shall be able to resolve the correct finger clustering i.e. which fingers are paired. The model we present Figure 8 is one possible specification that can handle dynamic finger re-clustering and resolve possible inconsistencies of the previous model and is divided into four different parts:

- The first part is the same as the Pinch interaction model presented in Figure 5 (augmented only with the transition remainingPinch).
- The second part is in charge of managing clustering. When the model receives a "toucheventf_move", the related pinching fingers trajectories are analyzed to verify their match. If such is the case, the clustering ok transition is crossed and the pinching fingers are stored back in the PinchReady place; otherwise, they are put in the Re-Clustering place. From this point, the analysis is recursive as long as the PinchReady place is empty or a correct match for the finger trajectories is found which leads to four possibilities after taken a pair of pinching fingers from the PinchReady place: a match is found (two possibilities), no finger corresponds and either the PinchReady place is empty or not. In this last case, the finger clustering is let as is until a next "toucheventf_move" event is received.
- The third part is composed of meta-event listeners capable to monitor the state of particular transitions and places in the model.
- Once the re-clustering has been computed, the fourth part’s behavior is designed to re-locate all pinching fingers in their idle state setting the corresponding tokens in place PinchReady.

Gesture recognition

Formal description of multi-touch gestures is proposed in PROTON++ is based on regular expressions [28]. To enable such specifications, PROTON++ introduces directions (South, North…) to the touch events used by the gesture recognizer. A gesture is a sequence of finger movement which directions match its description. The ICO formalism addresses such specification even though it is not presented in this paper due to space constraints. The events represented in PROTON++ by means of regular expressions are described in the synchronized transitions in ICO. The touch direction attributed computed in PROTON++ by combining position associated to previous touch events past with the position of the current touch event is represented in ICO adding the same mechanisms in the low-level transducers.

The main advantage of ICO with respect to PROTON++ is that on one hand it makes explicit the various states the interaction techniques can be in and, on the other hand explicitly supports concurrency both in terms of fork and join. Such elements remain implicit in PROTON++ as interaction techniques are handled independently and it is even recommended to remove them at design time as they are not recognized by regular expressions1. However, for sake of readability of the models this direction management has not been represented in the models.

Adding Resilience to manipulation errors

Many studies such as [2] have highlighted limitations of touch manipulations and have considered solutions to overcome them ([2, 10]).

1 “Proton++ recognizes a gesture when the entire touch event stream matches a gesture regular expression. Each time a match is found, Proton++ executes the callback associated with the gesture expression and flushes the stream. Thus, with a single stream, Proton++ is limited to recognizing at most one gesture at a time.” Paragraph “SPLITTING THE TOUCH EVENT STREAM” Page 6 from[28]
However, less work has been published on false touch handling which is the purpose of this paragraph. While interacting and more frequently in case of turbulences, users may inadvertently and briefly touch the screen triggering false touches events.

Figure 8 - Pinch interaction model with dynamic re-clustering

In order to maintain high usability level and not display blinking feedback, interaction models shall be resilient to such manipulations errors. Figure 9 presents such a behaviour applied to the initial detection of a Pinch interaction technique. If a “touchevent_down” is received less than Xms after the second down, a temporary fusion is processed. This 3 fingers fusion is validated and sent to the 3 fingers pinch model after 5ms unless a “touchevent_up” is received within this time frame.

ANALYSIS
The Petri net formalism enables the use of several analysis capabilities provided by the community. Petshop tool is used to perform the invariant analysis directly on ICO models. CPN Tools together with the work of Silva et al. [43] is used to accomplish the structural analysis and reachability graph analysis to CPN models converted from ICO models. Invariants, standard properties (e.g. liveness) and properties based on patterns (e.g. consistency, precedence) were identified in the presented models but are not presented due to space constraints.

CONCLUSION AND PERSPECTIVES
This paper has presented how multi-touch interaction techniques 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 multi-touch interactions modelling. More precisely we demonstrated how dynamic instantiation of input devices (fingers), dynamic reconfiguration of interactions, fusion of multiple events, clustering (grouping of input devices involved in the same interaction) and explicit handling of true concurrency enable multi-touch interaction specification. Gesture recognition has only been mentioned even though it...
is easily manipulated thanks to the capability of ICO to handle complex tokens carrying values. The examples given have presented in detail how multi-touch interactions modelled with ICOs can cooperate in order to produce high-level events such as Pinch and TapAndHold meaningful for the interactive application.

This work belong to more ambitious research programme aiming at producing methods, tools and techniques for the engineering of multimodal and multi-touch 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 multi-touch interfaces to the area of safety critical systems.

ACKNOWLEDGEMENTS

This work is partly funded by Airbus under the contract CIFRE PBO D08028747-788/2008

REFERENCES


