Brace Touch: A Dependable, Turbulence-Tolerant, Multi-touch Interaction Technique for Interactive Cockpits

Philippe Palanque[1,2(✉)], Andy Cockburn[3], Léopold Désert-Legendre[1], Carl Gutwin[4], and Yannick Deleris[5]

[1] ICS-IRIT, Université Paul Sabatier, Toulouse 3, France
palanque@irit.fr
[2] Industrial Design, Technical University of Eindhoven, Eindhoven, The Netherlands
[3] University of Canterbury, Christchurch, New Zealand
andy@cosc.canterbury.ac.nz
[4] University of Saskatchewan, Saskatoon, Canada
gutwin@cs.usask.ca
[5] Airbus Operations, Toulouse, France
yannick.deleris@airbus.com

Abstract. A cockpit (also called a flight deck) is an interactive environment of an aircraft that enables both pilot and first officer to monitor and control the aircraft systems. Allowing the crew to control aircraft systems through display units by using a keyboard and cursor control unit is one of the main features in the new generation of cockpits based on the ARINC 661 standard. Aircraft manufacturers are now investigating the deployment of touch interactions in future cockpits and ARINC 661 standard (supplement 7) extends it for that purpose. While touch interactions have demonstrated benefits in terms of performance (from the user point of view), their dependability is an important issue that has not been addressed so far. This paper proposes an interaction technique for touch devices called Brace Touch that aims at increasing the dependability of touch interactions by providing solutions to address development, natural and operation faults.

Keywords: Human-computer interaction · Interaction techniques · Fault-tolerance · Dependability · Usability

1 Introduction

The evolution of cockpits in large civil aircraft has followed two different paths:

- Small increments/evolutions targeting identified problems or integrating new equipment (similar to existing technology) into an existing cockpit;
- Significant steps/evolutions resulting in complete re-design of the cockpit including control and displays. Examples of such evolutions include the glass cockpit (where large display units were included in the flight deck) and more recently interactive

A. Romanovsky et al. (Eds.): SAFECOMP 2019, LNCS 11698, pp. 53–68, 2019.
https://doi.org/10.1007/978-3-030-26601-1_4

cockpits compliant with ARINC 661 specifications [3] where interaction takes place through mouse-like input devices and keyboards.

In parallel with these evolutions, multi-touch interfaces have appeared in most environments including mobile technologies, flight entertainment systems, and consumer electronics. Such interfaces have demonstrated benefits related to the fact that the output device (screen) integrates input management (touch device) thus bridging the (usual) gap between input and output in user interfaces. These systems have also demonstrated benefits in terms of performance for triggering commands by exploiting multi-finger interactions that can reduce the number of unnecessary modes.

Changing interaction in cockpits brings a set of issues that already appeared when interactions in the cockpit evolved from "physical interactions" (by manipulating physical knobs and reading information on dials) to software controls mainly based on the ARINC 661 specification [3]. Often considered as reliable means to trigger commands, devices such as pushbuttons, rotators and safe-guarded physical buttons are now common in the flight deck. However, these devices generate significant weight load and bring maintenance issues in terms of augmentation of cockpit functions [6]. This is a key issue as these physical components are directly linked to the aircraft systems they control, resulting in the fact that evolutions (of the aforementioned systems) are likely to require modifications to the components themselves or their organization in the cockpit. This is extremely costly as tuning part of the cockpit is likely to have broad implications for the aircraft and its operation [24]. Local changes might affect performance, and ultimately require adaptation of procedures and flying crew training going far beyond the system itself. Bringing touch interactions to commercial cockpits offers potential advantages to pilots, airlines, and aircraft manufacturers. Pilots may benefit (beyond the touch advantages presented above) from their familiarity with touch interactions on personal devices. Airlines and manufacturers could benefit from reduced hardware installation complexities, easier maintenance and reduced training. Replacing hard-wired physical controls with touchscreens could therefore ease development, facilitate upgrades, reduce weight, and improve pilot interaction as argued in [8]; and this is why AEEC produced a supplement of ARINC 661 for touch [4].

Providing multi-touch interactions for the command and control of civil aircraft would be another significant evolutionary step as far as cockpits design is concerned. However, together with the benefits presented above, multi-touch interfaces bring a set of issues that are still to be solved prior to making them "certifiable" and thus deployable for command and control of (safety) critical interactive systems. These issues affect multiple properties such as *reliability*, *fault-tolerance*, and *usability*.

Our previous work [22] and [21] mostly addressed the modeling aspects of multi-touch interactions and their use for cockpit applications to deal with development faults by providing a dedicated formal description technique based on high-level Petri nets [20]. This paper builds on that previous work and focuses on the dependability aspects of multi-touch interactions from a holistic point of view, addressing development faults, natural faults, and operation faults (according to the taxonomy from [6]). To this end, we propose a new interaction technique called Brace Touch (introduced in [12]) where the empirical evaluation of its usability is demonstrated. This interaction

technique allows flying crew to interact with touchscreens even under turbulence [11], which one of the main usability problems for touch interactions in cockpits, together with muscular fatigue and postures, as studied by Airbus human factors experts [8].

The remainder of this paper is organized as follows. The next section describes the underlying hardware, software, and interaction techniques for touchscreens. Section 3 identifies the dependability aspects of touch interactions. Section 4 presents the design and the principles of Brace Touch interaction together with a precise description of its behavior. Section 5 presents our proposal for addressing the dependability of Brace Touch interactions, including development, operation, and natural faults. Section 6 concludes the paper and highlights future directions.

2 Touch Interactions

This section introduces the usability and user experience (UX) aspects of touch interactions before focusing on touch device hardware, software and interaction aspects.

2.1 Usability and UX (Operations)

As presented above and argued in [36] (much before the advent of multi-touch interactions on personal and mobile devices), touch interactions provide the advantage of bridging the gap between input and output allowing users to interact without any artificial input device such as stylus or mouse. Interaction thus does not occur remotely but directly at the fingertip of the users, and early work such as [26] has demonstrated that touch screens can provide better performance for task completion. Such benefits do not come without disadvantages. First, current user interfaces are designed for interaction with mice or trackpads that provide higher levels of precisions that the so-called "fat finger". Fingers occlude small targets during selection and this reduced precision results in higher error rates on small targets. Beyond fingers, hands and arms increase occlusion on large touch screens. These issues have been at the center of recent research in human-computer interaction and solutions have identified new touch interaction techniques such as offering a fixed cursor offset [34], enlarging the target area [32], and providing on-screen widgets to facilitate selection [1].

This paper proposes a similar approach presenting the Brace Touch interaction technique designed to guarantee a reasonable level of usability even in the context of moderate or severe turbulence. In other work on compensating for vibration during touch interaction, [11] has shown that users stabilized their hand by resting on the bezels, i.e., the non-interactive part of the screen, in order to select targets with accuracy. However, as the hand is stabilized on the bezel only a small proportion of a large display can be reached. The Brace Touch interaction aims at increasing accuracy and reducing errors during turbulence while allowing interaction with the entire touchscreen.

2.2 Hardware

A touchscreen consists of a pointing device (touch sensor) combined with a display (e.g., LCD or CRT). The pointing device is responsible for capturing (X,Y) coordinates on the touch sensor and dispatching this information to the software component (exploiting a dedicated driver). There are multiple touch screen technologies but this paper focusses on the four main categories available on the market (beyond research prototypes): resistive, capacitive, surface acoustical wave (SAW) and infrared (IR). The interested reader can find a full review on touch technologies in [40].

Resistive technology consists of two conductive layers separated by an isolating layer (pierced with very small holes in a grid manner). On finger contact, the two layers get in contact at the closest location of a hole, creating a new voltage. This technology is cheap, contaminant-proof and works with any pointed object (not only a finger). However, it has poor screen clarity and does not support multi-touch. Moreover, such technology requires pressure, which may interfere with user interaction and performance.

Capacitive technology uses the user's body capacitance to detect contact on the screen. This technology is largely used on smartphones since Apple's first iPhone in 2007. It offers excellent clarity and supports multi-touch, but only detects body contact.

Infrared technologies consist of IR LEDs and photodetectors organized as a grid. The IR light is stopped by an opaque object contacting the screen. IR technology offers excellent screen clarity, supports multi-touch, and is very robust to the environment. However, such technology remains expensive.

Surface acoustic wave technology uses piezo transducers that emit and receive ultrasonic Raleigh waves along the bezels of the screen. On contact, the finger partially absorbs the wave which results in a temporary loss of amplitude in the received wave. This technology is cheap and offers excellent visual quality. However, it requires the user to press (rather strongly) on the screen (thus potentially reducing user performance) and it only supports two contact points.

This multiplicity of technologies demonstrates the maturity of the domain and the fact that diversity can be addressed at the hardware level to tolerate and possibly remove hardware faults. For instance, on the input side, it would be possible to combine a layer of capacitive technology on top of a resistive touchscreen. The main touchscreen manufacturers do not provide information about the frequency of hardware transient failures, but 10^{-5} is informally put forward by manufacturers. Such a combination would be a path towards deployment of tactile screens for Development Assurance Levels (DAL) B or C applications [13] by bringing together diversity, redundancy and segregation at the hardware level.

2.3 Software

Even though hardware can significantly vary from one touch screen to another, most programming toolkits offer the same low level events and information: a location (X,Y) on the screen, an event type *Down*, *Move*, or *Up* and a touch id to identify continuous *Move* events. This id allows identifying that the second move event received has been performed by the same finger as the first move. Apart from this common basis, particular hardware can enrich the touch information with physical data such as pressure,

finger contact area, or finger orientation. All this information may be used to design touch interaction techniques (as presented in Sect. 2.4).

Many dedicated programming toolkits exist for the development of multi-touch interactive applications, including libraries from hardware platform manufacturers such as Google, Microsoft, or Apple. Such toolkits usually support the user interface guidelines defined by the manufacturer and are more or less in line with the ISO standard on touch devices [25]. The number of users and the iterative correction of defects are typical arguments put forward to argue for the reliability of this layer. However, despite some work on the use of formal methods for input device driver description [2], OS manufacturers do not provide information about the reliability means used in the development of that layer.

2.4 Touch Interaction

Touch interaction techniques are highly dependent on the low-level events produced by the hardware. As seen in the previous section, most platforms only provide touch location, the event type, and an identification number of the finger (id). These events enable designers to build temporal interactions (e.g., dwelling), spatial interactions [7], and spatio-temporal interactions such as gestures [27]. Other interactions rely on hardware capturing particular physical data such as pressure [12], finger contact area [23], or finger orientation [41]. Finally, finger identification consists in linking a touch event to a particular finger; this problem has proved difficult even though some research contributions are promising using geometrical analysis of touch points [5], using additional camera input [15], biometric data [37], or additional hardware worn on the fingers [28].

These elements provide the basic underlying language on top of which touch interaction techniques have to be built. For instance, a horizontal movement of the finger on the surface of the screen will trigger a higher level event called a *swipe* in most platforms. This *swipe* event is only triggered if the quantity of horizontal movement is large enough and if no additional fingers have been pressed during the movement. Only work from [22] has addressed the formal definition of touch interaction techniques that are usually crafted jointly by user interface designers and developers. Interaction techniques are thus usually of low reliability and exhibit faults at operation time. Detection of development faults is only addressed by testing which cannot be exhaustive due to the very nature of interactive systems [29]. Beyond that, the inner nature of interaction techniques might be more or less fault-tolerant (as shown in Fig. 1). In case of a hardware failure of a line on the touchscreen hardware grid, the swipe in Fig. 1(a) might not be recognized even though the operator has correctly performed the gesture. In Fig. 1(b) the swipe gesture is now an upside-down V shape that could still be recognized even in the presence of several hardware touch screen lines failures. However, these more complex gestures come with a cost in terms of usability (more movements and thus more time) and their use must be the result of a thorough analysis of the usability-dependability trade-off, such as the ones presented in [17] for standard non-touch interaction techniques.

Fig. 1. Tolerance of *swipe* touch interaction techniques to hardware faults (loss of a line in the hardware grid) – (a) non tolerant (b) tolerant

3 Fault Model for Touch Interactions

As for any kind of computing systems, multi-touch interactive systems can be subject to faults at development and operation time but may also be subject to natural faults (such as bit flips due to cosmic rays). Interactive cockpits are a specific kind of command and control system and, as such, the entire taxonomy of faults presented in [6] applies. The fault model considered in the current paper does not include malicious faults because, during operation, the cockpit can be considered as a closed world and development is carefully monitored.

Our research work aims at contributing to the dependability of touch interaction by considering development, natural, and operation faults. Due to space constraints, the current paper focusses on operation faults for touch interactions but we present in this section how to address other faults.

Addressing Development Faults. We propose to address development faults by using formal description techniques and dedicated verification mechanisms. However, due to the very nature of interactive systems, specific methods are required that are able to handle:

- *Concurrency*: for instance to represent multiple concurrent interactions with multiple fingers on the touch screen;
- *Dynamic instantiation of objects*: for instance, when fingers are added and removed from the touch screen, their reference must be captured by models;
- *Quantitative time*, as most interaction techniques include temporal evolutions: for instance, the time between two touch events required to trigger a double-tap higher level event;
- *Large number of states*: for instance, current requirements for touch screen in aircraft cockpit mandate the management of up to 20 fingers on a given touch screen. Such a large number of fingers make it possible to use complex interaction such as finger clustering [21] exhibiting a very large state space.

For this reason, we propose to use a formal description techniques called ICOs [31] that covers these needs and has been used for describing in a complete and unambiguous way all the components of the previous generation of interactive cockpits [9].

Addressing Natural Faults. We consider here a similar fault model as the one presented in [16] with a focus on erroneous control (transmission of a different action from the one done by crew members – e.g., sending a *swipe* event while the user has performed a *long tap*) and inadvertent control (transmission of an action without any crew member action). These faults can be addressed by extending touch technologies

with self-checking capabilities as was done with interactive cockpits [16] but this is not presented here due to space constraints. In addition, the diversity and redundancy of hardware platforms as presented above would increase tolerance to natural faults.

Addressing Operation Faults. In aircraft cockpits, operation faults typically fall in the classification of human errors either being intentional (called violation) or non-intentional (called mistakes, lapses, or slips depending on their type [35]). While such errors may have multiple sources, some of them are induced by the user interface design itself [38], but they can result from the operation environment too. For touch interactions, there is a need for safeguards against undesirable and potentially dangerous environmental conditions, such as turbulence, that might be precursors for each error type presented above. This paper proposes an interaction technique that aims at reducing operation faults triggered by turbulence.

4 Brace Touch to Improve Dependability of Touch Interactions

4.1 Principles of Brace Touch

The primary design intention of Brace Touch is to allow users to mechanically stabilize touch interactions by concurrently placing multiple fingers onto the display. Selections and other interactions are then completed by the user issuing a recognizable gesture while the other stabilizing contacts are maintained on the display. For example, Fig. 2 shows the index finger selecting an item by tapping while all of the other digits of the hand are pressed onto the display.

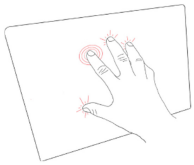

Fig. 2. Illustration of the brace touch interaction concept

Brace Touch uses the multi-touch capabilities of contemporary touchscreens to give users two alternative ways of completing selections. First, when the interaction environment is stable (i.e., no turbulence), users can select items by simply tapping on them with a single finger, which allows convenient touchscreen interaction in a manner similar to that used on current mobile devices. Second, during turbulence, braced selections can be made when certain selection criteria are satisfied. These selection criteria involve waiting until at least four concurrent contacts are registered on the touchscreen, then selecting the item (if any) beneath the last-placed contact when it is removed from the display. In this way, the user can place four fingers of one hand (e.g., all but the index finger) onto the display – thereby stabilizing their hand on the display– then complete a gesture on the target using their index finger (a fifth contact) to select

it. Subsequent selections can be made by dragging the four initially placed contacts across the display, then gesturing again with the index finger – there is no need to lift all fingers off the display when completing a series of selections.

Our experimental validation of Brace Touch (results briefly described in Sect. 5) compared user performance with three different variants of the terminating selection gesture: double tap, dwell (a long press), and force-press. As double-tap is the most usable (better efficiency and higher satisfaction), we detail this technique in the following sections).

4.2 Hardware and Software Architecture of Brace Touch Interaction

Figure 3 shows the general software architecture to manage touch events in order to provide brace touch interaction. The architecture is mapped onto the generic ARCH architecture for interactive systems [10] (left-hand side of Fig. 3). As a result, the hardware and drivers belong to the physical interaction level, finger management and brace interaction models to the logical interaction level. We present the formal description of each element of this logical interaction level using the ICO description technique [31].

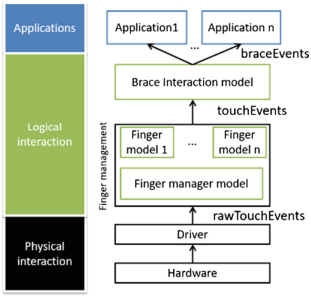

Fig. 3. High-level software architecture of brace interaction

For each user action, the hardware driver produces raw touch events with basic information (e.g., the event type and a touch id). The finger manager handles those events and manages creation and destruction of finger models. The finger model processes low-level events provided by the hardware layer and offers data consultation and update services. The brace interaction model handles interaction logic from the finger manager and produces higher-level events (brace touches) dispatched to interested applications.

4.3 Behavioral Description of Brace Touch Using ICOs

Informal Description of the ICO Formal Notation. The ICO (Interactive Cooperative Objects) notation is a formal description technique devoted to interactive systems. Using high-level Petri nets [20] for dynamic behavior description, the notation also

relies on an object-oriented approach (dynamic instantiation, classification, encapsulation, inheritance and client/server relationships) to describe the structural or static aspects of systems. ICO notation objects are composed of four components: a cooperative object for the behavior description, a presentation part (i.e. Graphical Interface), and two functions (activation and rendering) describing the links between the cooperative object and the presentation part. The interested reader can find more information in [31]. ICO addresses all the issues presented in the previous sections including representation of dynamic instantiation of input devices (fingers) which is not addressed in other modelling techniques. For instance, ICO uses the capabilities of Petri nets to describe tokens' creations which are used in models to represent input devices such as fingers on a touchscreen. In the following sections, we will use the following formalism to describe ICO models: **places**, *events*, and transitions.

Finger Behavioral Model. The finger model handles touch information through the *Info* place (center of Fig. 4). The model provides accessors services to touch data through *getTouchInfo* which is accessible by other models to get information about a given finger. The information of the finger can be modified using the *update* service called by the low level architecture to store a new value for the finger. Finally, the *terminate* service handles finger model removal (triggered when the finger is not in contact with the touchscreen anymore).

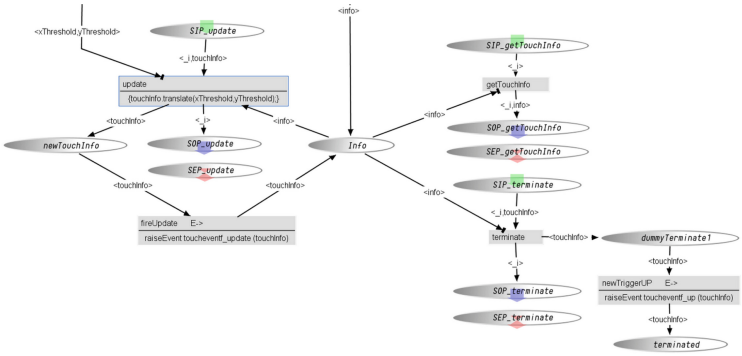

Fig. 4. ICO behavioral model of a finger using ICO notation

Finger Manager Behavioral Model. The finger manager is a model responsible for managing the set of fingers currently on the touch screen. That model received raw events from the platform and abstracted them away in meaningful information in terms of interaction. Figure 5 shows an abstract model of the behavior of the finger manager (the full ICO model cannot be presented due to space constraints). In this view, **places** are represented with ovals, transitions with rectangles, event transitions by rectangles with italic text, produced *events* in italic font alongside the producing transition, and conditions of transition activation under the transition name, separated by a horizontal line.

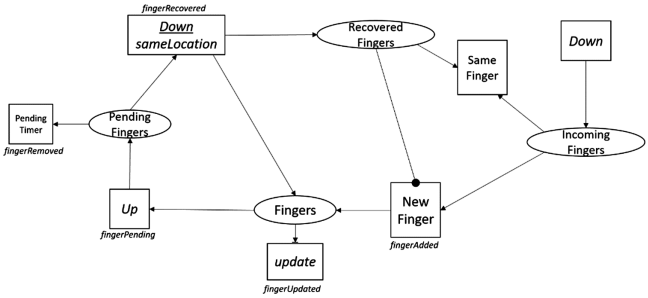

Fig. 5. Abstract behavioral model of the finger manager

In Fig. 5, place **Fingers** stores the references to the finger models (one per finger currently present on the touchscreen). As Brace interaction relies on double tap for target selection, the finger used to perform this interaction will leave (for a short period of time) the surface of the touch screen and will come back at the same location. According to the finger model, leaving the surface would require removing the finger from the pool of fingers. The Brace model thus handles this short disappearance by storing fingers in the **Pending Fingers** place. In order to disambiguate an *Up-Down* from the same finger with an *Up* and *Down* from two different fingers, we chose a temporal and geographical logic. Thus, on finger *Up*, the reference to the finger model is set in place **Pending Fingers** for a duration defined in the PendingTimer transition. If a *Down* event is received before the time has elapsed, its location is compared with the location of the finger when *Up* occurred. If the location is close enough (a parameter in the model), the same finger model is updated and set back into **Fingers** and a *fingerRecovered* event is raised (*Down/sameLocation*). A copy of the reference of that finger model is also stored in **RecoveredFingers**.

Down events correspond to the arrival of a new finger on the touchscreen. When this occurs, a new finger model is instantiated (by the finger manager) and its reference is stored in **IncomingFingers**. If the *Down* event corresponds to a finger that was in the pending state, the information stored in **IncomingFingers** is filtered with SameFinger, thus preventing the creation of a new finger model. Finally, on update, the finger model is updated and *fingerUpdated* event is raised so the information (e.g., the location on the screen) is updated in the corresponding finger model.

Figure 6 shows the finger manager model pending process using the ICO notation. PendingTimer, timed transition is set to 500 ms which corresponds to the maximum time allowed to the user to perform a double tap. Finally, the geographical disambiguation logic consists in verifying that the *Down* event received lies within 30px from the previous *Up* event location (see shapeProcessing transition performing the proximity calculation).

Brace Touch Interaction Behavioral Model. Figure 7 shows an abstract view of the Brace Touch interaction behavior in a similar way as we presented the finger manager in Fig. 5.

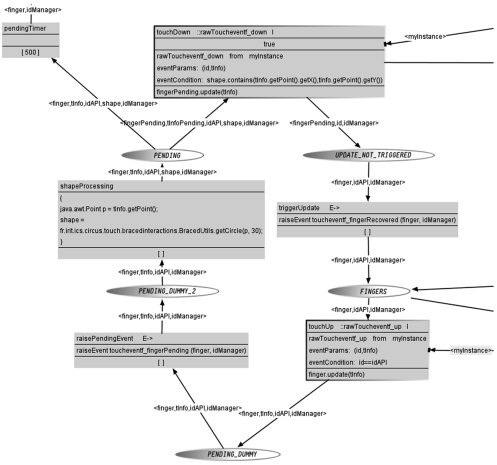

Fig. 6. Part of the finger manager behavior filtering new fingers from returning fingers

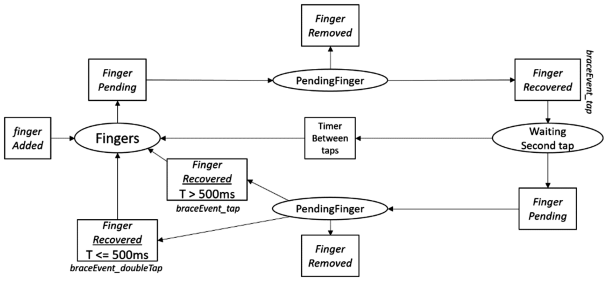

Fig. 7. Abstract model of Brace Touch interaction behavior

The Brace Touch Interaction model listens to the events produced by the Finger Manager namely *fingerAdded*, *fingerPending*, *fingerRemoved* and *fingerRecovered* events, the production of which was detailed in the previous section.

On *fingerAdded*, the reference of the created finger model is stored in place **Fingers.** According to Brace Touch design [12], a double tap is specified as two consecutive taps within 500 ms, a tap being a consecutive *Up* and *Down* with the same finger on the same location (spatial threshold of 30 pixels). Thus, on finger *Up* and *Down*, the finger manager produces respectively *fingerPending* and *fingerRecovered*, unless the finger does not touch the screen again, in which case a *fingerRemoved* event is received. On fingerRecovered, the *braceEvent_tap* is raised and the reference to the finger model is stored in **WaitingSecondTap**.

If a second tap is not initiated within the time frame specified in TimerBetweenTaps, the double tap interaction is exited and the reference to the finger model is stored back in **Fingers**. The second tap interaction has the same behavior as the first tap, apart from the validation of the double tap interaction: if the time between the first tap and the second

tap is less than 500 ms, *braceEvent_doubleTap* is raised. Otherwise, the second tap is considered as a single tap following a first tap.

Due to space constraints we cannot present the ICO model of brace interaction but its main features have been highlighted. It is important to note that these models can be analyzed using knowledge from Petri nets theory, such as computing locks and traps as well as P and T invariants [18]. These verification techniques allow interaction designers to assess the correctness of a given model (absence of deadlock) and also evaluate the compatibility between models (e.g., the events produces by a model are consumed by an upper model). The formal analysis performed by the development platform called PetShop [18] supports the detection of development faults.

5 Dependability and Usability of Brace Touch

While a formal notation was presented in previous section to deal with software development faults of interaction techniques in general and Brace Touch interaction in particular, dealing with natural and operational faults is required to bring such concepts into real operation in aircraft cockpits. We have briefly mentioned diversity, redundancy and segregation in hardware technologies for touch devices as a mean to address hardware faults.

Natural faults could affect software (via bit flips, for example) but also operators (via vibration of the touch device). As for natural software faults, self-checking techniques for interactive widgets and interaction techniques as presented in [39] could support detection and removal of faults in touch interactions. However, touch interactions are more sensitive to natural faults such as turbulence that will trigger unintentional operational faults. Brace Touch interaction aims at reducing the effect of turbulence on operational faults for touch interactions.

However, as presented in [17] the introduction of dependability mechanisms might affect negatively usability of the interactive system. It was thus important to assess the impact on the overall performance of braced interaction to ensure that the technique does not slow down operation. This is demonstrated in Fig. 8: in vibrating conditions, braced interaction with selection using double-tap has a similar (though slightly slower) performance to non-braced interactions in non-vibrating conditions.

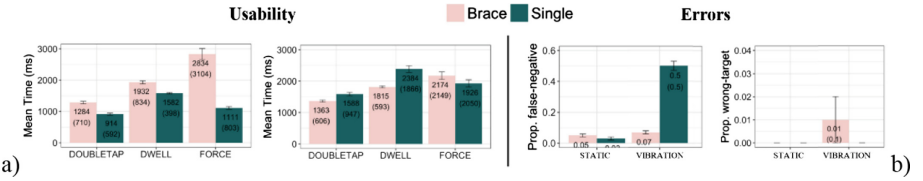

Fig. 8. Target selection times with Braced and Single-Finger selections in static (left) and vibrating (right) conditions [12].

In [12], Cockburn et al. report results of an experiment examining three different forms of braced selections in turbulent and static environments. Turbulence was simulated using a Mikrolar motion platform, with a seat and touchscreen mounted on the platform. The three bracing variants differed in the gesture used for completing a selection: doubletap, dwell (long press) and force threshold (not considered further in this paper). Participants also completed selections using traditional single-finger tap-to-select modalities, with this data serving as a baseline comparator for the braced selection methods. Additionally, in one portion of the experiment, participants were allowed to choose whether to complete selections using braced touch (placing multiple fingers on the display) or normal hand postures (using a single finger without bracing, terminated with a doubletap, dwell or force-press).

Figure 8 summarises the usability study of Brace Touch looking at performance (a) and user errors (b). Selection time results using the three bracing methods and equivalent single-finger selections in static (pink bar) and turbulent (bleu bar) settings. As expected, in static settings single-finger selections are faster than braced selections, but when vibration is present, braced selections were significantly faster and more accurate than single-finger selections. In tasks where participants were free to choose whether or not to use braced postures, 88% of double-tap users chose to use bracing when vibration was present, compared to only 19% when static. This suggests that users were aware of the benefits provided by bracing and appropriately chose to use it to overcome interaction problems stemming from vibration.

Two types of errors were analysed: false-negative selections, in which subjects tried but failed to complete a selection; and wrong-target selections, in which a target other than the intended one was selected. Figure 8(b) summarises the results of errors from the user study. The histogram highlight the strong tolerance to vibrations of Brace Touch with respect to false negatives. Value of wrong target selection (not as good as false negatives) was influenced by the tasks performed by the users which were related to aircraft operations with which they were not familiar. In [12], Cokburn et al. show that for the later tasks, wrong target selection errors with Brace Touch disappeared and that in 87.3% of selections, subjects elected to use braced selections during vibrations.

According to Avizienis et al.'s taxonomy [6], as far as touch screens are concerned, vibrations would belong to natural faults triggering non-intentional human-made faults during operation, and would correspond to slips in the human error classifications [35]. Looking at the interaction itself, the operator decides intentionally to interact with the touch screen and to use (preferably) Brace Touch interactions in cases of turbulence [12]. However, while interacting, the user might unintentionally make errors that will prevent Brace Touch interaction from being performed. For instance, in Brace, the index finger is meant to be the finger used for performing the double-tap while the other fingers are constantly touching the touch device. Vibrations might make other fingers (especially the little finger) bounce and thus trigger unintentional double-taps. We have extended the behavior of Brace Touch to detect the position of index fingers (which can be deduced from the relative position of the other fingers) and discard spurious double-taps triggered by the other fingers. Due to space constraints, we do not present the ICO model here, but the approach is the same as the one presented in Brace Interaction model.

6 Conclusion and Perspective

New interactive technologies are making their way into command and control positions of safety critical systems. This is due to the new type of work and new type of data that operators have to manipulate, but also because those technologies are now pervasive and used in everyday life. This paper presented a contribution to the dependability of touch interactions to be deployed in real life contexts such as aircraft cockpits.

We have presented a formal description technique dedicated to interactive system description that addresses the problem of development faults in interaction techniques. We used that formal description technique to model a new touch interaction technique called Brace Touch that is able to avoid operational natural faults. We also shown that Brace Touch does not jeopardize the usability of touch interactions, providing good performance, recall, and user satisfaction.

This work cannot be separated from design aspects of interactive systems that include how to present information to the operators, and more precisely, the size and the colors of objects. Previous work [14] has highlighted that vibrations affects perception and thus there is a need to address the entire interaction loop (perception, cognition, and action). This is what will be done in future studies that could lead to user interface guidelines for touch interfaces in transportation. It is important to note that previous work on model-based approaches for usability engineering [33] and interaction reconfigurations [30] provide some bases to address that problem.

References

1. Albinsson, P.A., Zhai, S.: High precision touch screen interaction. In: Proceedings of ACM CHI Conference, pp. 105–112 (2003)
2. Accot, J., Chatty, S., Maury, S., Palanque, P.: Formal transducers: models of devices and building bricks for the design of highly interactive systems. In: Harrison, M.D., Torres, J.C. (eds) Design, Specification and Verification of Interactive Systems 1997. Eurographics, pp. 143–159. Springer, Vienna (1997). https://doi.org/10.1007/978-3-7091-6878-3_10
3. ARINC 661 Cockpit Display System Interfaces to User Systems. ARINC Specification 661. Airlines Electronic Engineering Committee (AEEC) (2002)
4. ARINC 661. Cockpit display system interfaces to user systems. ARINC Specification 661, supplement 7 (April 2019). Airlines Electronic Engineering Committee (AEEC) (2019)
5. Au, O.K.C., Tai, C.L.: Multitouch finger registration and its applications. In: ACM CHI Conference, pp. 41–48 (2010)
6. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
7. Bailly, G., Lecolinet, E., Guiard, Y.: Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces. In: ACM CHI Conference, pp. 591–594 (2010)
8. Barbé, J., Chatrenet, N., et al.: Physical ergonomics approach for touch screen interaction in an aircraft cockpit. In: Conference on Interaction Homme-Machine (IHM), pp. 9–16. ACM DL (2012)

9. Barboni, E., Conversy, S., Navarre, D., Palanque, P.: Model-based engineering of widgets, user applications and servers compliant with ARINC 661 specification. In: Doherty, G., Blandford, A. (eds.) DSV-IS 2006. LNCS, vol. 4323, pp. 25–38. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69554-7_3
10. Bass, L., et al.: The arch model: Seeheim revisited. In: User Interface Developers' Workshop, April 1991
11. Cockburn, A., et al.: Turbulent touch: Touchscreen input for cockpit flight displays. In: Proceedings of ACM CHI Conference, pp. 6742–6753 (2017)
12. Cockburn, A., et al.: Design and evaluation of brace touch for touchscreen input stabilisation. Int. J. Hum.-Comput. Stud. **122**(21–37), 7 (2019)
13. DO-178C/ED-12C, Software Considerations in Airborne Systems and Equipment Certification, published by RTCA and EUROCAE (2012)
14. Dodd, S., Lancaster, J., Miranda, A., Grothe, S., DeMers, B., Rogers, B.: Touch screens on the flight deck: the impact of touch target size, spacing, touch technology and turbulence on pilot performance. In: Proceedings of the HFES Annual Meeting, vol. 58, no. 1, pp. 6–10 (2014)
15. Ewerling, P., Kulik, A., Froehlich, B.: Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions. In: ACM TEI Conference, pp. 173–182 (2012)
16. Fayollas, C., Palanque, P., Fabre, J-C., Martinie, C., Déléris, Y.: Dealing with faults during operations: beyond classical use of formal methods. In: [19], pp. 549–575 (2017)
17. Fayollas, C., Martinie, C., Palanque, P., Deleris, Y., Fabre, J.-C., Navarre, D.: An approach for assessing the impact of dependability on usability: application to interactive cockpits. In: IEEE European Dependable Computing Conference (EDCC), pp. 198–209 (2014)
18. Fayollas, C., Martinie, C., Palanque, P., Barboni, E., Fahssi, R., Hamon, A.: Exploiting action theory as a framework for analysis and design of formal methods approaches: application to the CIRCUS integrated development environment. In: [19], pp. 465–504
19. Weyers, B., Bowen, J., Dix, A., Palanque, P. (eds.): The Handbook of Formal Methods in Human-Computer Interaction. HIS. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51838-1
20. Genrich, H.J.: Predicate/transitions nets. In: Jensen, K., Rozenberg, G. (eds) High-Levels Petri Nets: Theory and Application. Springer, Heidelberg, pp. 3–43 (1991). https://doi.org/10.1007/978-3-642-84524-6_1
21. Hamon-Keromen, A., Palanque, P., Deleris, Y., Navarre, D., Barboni, E.: A tool-supported development process for bringing touch interactions into interactive cockpits for controlling embedded critical systems. In: HCI in Aeronautics (HCI'Aero), pp. 25–36. ACM DL (2012)
22. Hamon-Keromen, A., Palanque, P., Silva, J.-L., Deleris, Y., Barboni, E.: Formal description of multi-touch interactions. In: ACM Engineering Interactive Computing Systems, pp. 207–216 (2013)
23. Harrison, C., Schwarz, J., Hudson, S.E.: TapSense: enhancing finger interaction on touch surfaces. In: Proceedings on ACM UIST Conference, pp. 627–636. ACM (2011)
24. Hutchins, E., Lauwsen, T.: Distributed cognition in an airline cockpit. In: Engeström, Y., Middleton, D. (Eds) Cognition and Communication at work. Cambridge University Press, Cambridge (1996)
25. ISO 9241-9:2010 Usability - Part 9: Requirements for non-keyboard input devices (2010)
26. Karat, J., McDonald, J., Anderson, M.: A comparison of selection techniques: touch panel, mouse, keyboard. Int. J. Man-Mach. Stud. **1**, 73–92 (1986)
27. Kin, K., Hartmann, B., Agrawala, M.: Two-handed marking menus for multitouch devices. ACM Trans. Comput.-Hum. Interact. (TOCHI) **18**(3), 16 (2011)

28. Marquardt, N., Kiemer, J., Ledo, D., Boring, S., Greenberg, S.: Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. In: Proceedings of ACM TEI, pp. 21–30 (2011)
29. Memon, A.M.: An event-flow model of GUI-based applications for testing. Softw. Test. Verif. Reliab. **17**, 137–157 (2007)
30. Navarre, D., Palanque, P., Basnyat, S.: A formal approach for user interaction reconfiguration of safety critical interactive systems. In: Harrison, M.D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 373–386. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87698-4_31
31. Navarre, D., Palanque, P., Ladry, J.F., Barboni, E.: 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 (2009)
32. Olwal, A., Feiner, S.: Rubbing the fisheye: precise touch-screen interaction with gestures and fisheye views. In: Conference Supplement of UIST 2003, pp. 83–84 (2003)
33. Palanque, P., Barboni, E., Martinie, C., Navarre, D., Winckler, M.: A model-based approach for supporting engineering usability evaluation of interaction techniques. In: Conference on ACM Engineering Interactive Computing Systems (EICS 2011), pp. 21–30 (2011)
34. Potter, R.L., Weldon, L.J., Shneiderman, B.: Improving the accuracy of touchscreens: an experimental evaluation of three strategies. In: Proceedings of CHI 1988, pp. 27–32 (1988)
35. Reason, J.: Human Error. Cambridge University Press, Cambridge (1990)
36. Shneiderman, B.: Touchscreens now offer compelling uses. In: Sparks of Innovation in Human-Computer Interaction. Ablex, Norwood (1993)
37. Sugiura, A., Koseki, Y.: A user interface using fingerprint recognition: holding commands and data objects on fingers. In: Proceedings of ACM UIST Conference, pp. 71–79 (1998)
38. Stanton, N., et al.: Predicting design induced pilot error using HET (Human Error Template) – a new formal human error identification method for flight decks. J. Aeronaut. Sci. **110**, 107–115 (2006)
39. Tankeu-Choitat, A., Navarre, D., Palanque, P., Deleris, Y., Fabre, J.-C., Fayollas, C.: Self-checking components for dependable interactive cockpits using formal description techniques. In: IEEE Pacific Rim Dependable Computing Conference, pp. 164–173 (2011)
40. Walker, G.: A review of technologies for sensing contact location on the surface of a display. J. Soc. Inf. Disp. **20**(8), 413–440 (2012)
41. Wang, F., Cao, X., Ren, X., Irani, P.: Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In: Proceedings of ACM UIST Conference, pp. 23–32 (2009)