Extending and validating *gestUI* using Technical Action Research

Otto Parra
Computer Science Dept.
Universidad de Cuenca, Ecuador
PROS Research Centre
Universitat Politecnica de Valencia
otpargon@posgrado.upv.es

Sergio España
Dept. of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
s.espana@uu.nl

Jose Ignacio Panach
ETS d'Enginyeria.
Dep. d'Informatica
Universitat de Valencia
Valencia, Spain
joigpana@uv.es

***Abstract*—gestUI is a model-driven method with tool support to define custom gestures and to include gesture-based interaction in existing user software system interfaces. So far, gestUI had been limited to the definition of the same gesture catalogue for all users of the software system. In this paper, we extend gestUI to permit individual users to define their own custom gesture catalogue and redefine some custom gestures in case of difficulty in using or remembering them. After extending gestUI, we applied technical action research from the FP7 CaaS project's Capability Design Tool with the aim of assessing its acceptance in an industrial setting. We also analysed its perceived ease-of-use and usefulness and gestUI's desirability level and user experience. The study shows that the tool can help improve the definition of custom gestures and the inclusion of gesture-based interaction in user interfaces of software systems.**

Keywords—Model-driven development; gesture-based interaction; human-computer interaction; technical-action research; user experience.

I. INTRODUCTION

At the present time, end-users work with different devices (e.g. mobile phones, tablets and notebooks) which provide a wide array of interaction mechanisms (e.g., gesture-based, gaze-based, voice-based interaction) [1], including the widely used gesture-based interaction. Aimaiti, N. et al.[2] argue that this type of interaction provides a new way for people to interact with devices and has been responsible for great advances in the history of human-computer interaction. Many software systems currently include predefined touch-based gestures (e.g. tap, double tap, swipe) for users to apply on user interfaces for their daily tasks[3]. However, interfaces that support gesture-based interaction using custom gestures have been reported to be more challenging to implement and test than traditional mouse and pointer interfaces [4]. Some of the challenges (e.g. complexity, cost, expertise required) can be tackled through a model-driven development (MDD) approach [5][6]. Using MDD, gestures and gesture-based interaction can be modelled and software code that implements them can be generated automatically.

MDD has been fairly popular in the academic community [7] in recent years, and a number of different proposals have been offered to develop software systems. MDD begins with the model's definition and then obtains the source code to different kinds of software through model transformations. Some of the benefits reported when MDD is integrated in the software development process include: increased productivity, easier maintenance and documentation, interoperability, reusability, and portability [8].

gestUI [9] is a model-driven method for including gesture-based interaction in the development of user interfaces for software systems. gestUI permits stakeholders (e.g. software engineers, end-users) to focus on the key aspects of gesture-based interfaces; defining customized gestures and specifying gesture-based interaction. Up to the present, gestUI had been limited to the definition of the same gesture catalogue for all users of the software system, besides which the gestures in the gesture catalogue could not be redefined by the users.

With the aim of solving this situation, in this paper we extend gestUI to allow each user to define his/her own custom gesture catalogue to obtain one gesture catalogue per user. Additionally, each user can redefine a custom gesture if he/she has problems in sketching it or in remembering it. Redefinition can be done during the execution stage (runtime) of the software system.

In this paper, we also validate gestUI together with the new extension through Technical Action Research (TAR).TAR can be seen as a research method that starts from the opposite side of traditional research methods.TAR starts with an artefact, and then tests it under practical conditions by using it to solve concrete problems [10].

According to Wieringa [10], TAR is related with the use of an experimental artefact to help a client and to learn about its effects in practice. The artefact is experimental, which means that it is still under development and has not yet been transferred to the original problem context. In a validation process with TAR, the researcher uses an artefact (e.g. method and a tool) in a real-world project to help a client, or gives the artefact to others so they can use it assisted by the researcher [11].

Here we report the validation of gestUI in real-world conditions through TAR. During the validation process, we

978-1-5090-5476-3/17/$31.00 ©2017 IEEE

aimed to discover just how gestUI can help stakeholders (e.g. software engineers, end-users) to define custom gestures and to include gesture-based interaction in existing user interfaces. We also aimed to obtain practical interpretations of the system from industry practitioners. We use TAR in the context of the CaaS Project (FP7 ICT Programme Collaborative Project no. 611351). The main outcomes of CaaS are: (i) the Capability-Driven Development (CDD) methodology [11] and (ii) the CDD environment. The Capability Design Tool is a CASE tool in the CDD environment that supports capability modelling according to the CDD meta-model [12]. Everis, a multinational firm offering business consulting, as well as development, maintenance and improved information technology, collaborated in the evaluation. Everis's CaaS-project team is developing an e-government platform by applying the whole CDD methodology and environment.

The main contribution of this paper is twofold: (i) we have extended the original gestUI method [9] to make it more flexible and capable of addressing more complex forms of software development (we also specify the method components using the MAP formalism [13]); (ii) by means of a TAR approach, we study how gestUI works in real-world conditions when applied to a CASE tool. We report on a user evaluation that involves business consultants using gestUI to include gesture-based interaction in a user interface and then carrying out a modelling task by means of gestures. We base the empirical validation on well-known frameworks and techniques, such as the Method Evaluation Model (MEM) [14] to relate subjects' performance, perceptions and intentions, the User Experience Questionnaire (UEQ) [15] to measure user experience with gestUI, and Microsoft Reaction Cards (MRC) to obtain desirability level and user experience [16] with gestUI.

The paper is organized as follows: Section II includes a Background on some of the terms used in this paper. Section III describes Related Work. In Section IV we explain the extension of the gestUI method to include flexibility. Section V introduces the Capability Design Tool from the CaaS project. Section VI describes the validation using technical action research. Section VII explains the action research procedure. The analysis and interpretation of the results are given in Section VIII. Threats to validity are analysed in Section IX. Section X contains conclusions and future work.

II. BACKGROUND

A. Map representation

A process model expressed in intentional terms can be represented by a *map representation* system, which provides a representation mechanism based on a non-deterministic ordering of intentions and strategies to model the multi-faceted purpose of a system [13].
An *intention* is a goal that can be achieved by performing a process [17]. For example, the gestUI map in Fig. 1 has "*Define a gesture (number 2)*" and "*Include the gesture in a repository (number 3)*" as intentions. Additionally, each map has two special intentions called "Start" and "End", to respectively start and end the process.
A *strategy* is an approach, a manner to achieve an intention [17]. In Fig. 1, "*By storing the gesture (transition from number 2 to number 3)*" is a manner to "*Include a gesture in the repository*" in a context of gesture-based interaction definition.
A map is graphically represented as a directed graph from start to stop. Intentions are represented as nodes and strategies as edges between nodes in the map representation.

B. Microsoft Reaction Cards

Product reaction cards are called Microsoft Reaction Cards since they were developed by Microsoft [18] as part of a "desirability toolkit" created to get the quality of desirability, a key component in user satisfaction[19].
Microsoft Reaction Cards (MRC) consist of a pack of 118 cards with 60% positive and 40% negative or neutral adjectives, from which subjects choose the words that reflect their feelings toward their interactive experience with a product [18]. Assessment based on Product Reaction Cards (PRC) has been recognized as one of the preferred methods for measuring the perceived desirability of visual designs [19].

C. User Experience Questionnaire

The main goal of the User Experience Questionnaire (UEQ) is to obtain a fast and immediate measurement of user experience of interactive products [20].The questionnaire format supports the user response to immediately express feelings, impressions, and attitudes that arise when they use a product [21]. The questionnaire consists of bipolar contrasting attributes on a seven-scale ranking. The subjects express their agreement with the attributes by ticking the circle that most closely reflects their impression. The seven-scale ranking is converted into a positive and a negative scale, where +3 represents the most positive and the -3 represents the most negative value [22]. The user experience questionnaire contains six scales with 26 items in total: attractiveness, efficiency, perspicuity, dependability, stimulation, novelty[23].

III. RELATED WORK

A. The role of gesture-based interfaces in Information Systems (IS) engineering

In the first part of this section we describe the role of the gesture-based user interfaces in information systems engineering.

Gesture-based interfaces can play two major roles in IS engineering, depending on whether we intend to incorporate this natural interaction into (i) CASE tools or (ii) the IS themselves. In the former case, the interest is to increase the efficiency of IS developers, whereas in the latter the aim is to increase the IS usability, especially in operations in the field, where the lack of a comfortable office space reduces mouse and keyboard ergonomics. In both cases, gesture-based interface development methods and tools are needed, some of which are described in this section:

Beuvens et al. describe UsiGesture [24], which allows a designer to integrate gesture-based interaction in an interface,

but lacks techniques to model, analyze or recognize gestures. The authors applied the method to develop a restaurant management tool. Guimaraes et al. [25] propose a method that includes requirements definition, design, implementation and evaluation. The authors apply it to creating a puzzle game. Nielsen et al. [26] describe a method with two variants (technology-based and human-based) and provides guidelines for the definition and selection of gestures, based on ergonomic principles. Bragdon et al. [27] describe GestureBar, which embeds gesture disclosure information in a familiar toolbar-based user interface. GestureBar's simple design is also general enough to be used with any recognition technique and to be integrated with any standard, non-gestural user interface component. The aim of Open Gesture is to provide inclusive interface designs that are usable by the elderly and disabled. Its authors (Bhuiyan et al. [28]) apply it to an interactive television project.

In this work we use gestUI [29], [30] which was designed to define custom gestures and to implement gesture-based interaction by applying model transformations and obtaining gesture-based interfaces.

B. *Applying technical action research in IS engineering*

Next, we describe some works related with the applicability of technical action research in the field of software engineering:

Morales-Trujillo et al. [31] describe the validation of a software engineering framework using Technical-Action Research and case study methods. They report that the combination of TAR and case studies was a successful experience and that it is a feasible resource for bridging the gap between academy and industry. Morali et al. [32] report the use of TAR to validate a method to specify the confidentially requirements in an outsourcing relation. They used CRAC++ to specify confidentially requirements that could be included in an outsourcing SLA. Abelein [33] describes the application of TAR to validate iPeople Case Study applying the User-Developer Communication-Large-Scale IT Projects (UDC-LSI) method. His evaluation revealed the positive effect of the UDC-LSI method on effectiveness and efficiency. Antinyan et al. [34] report a complementary empirical method for validating software measures. The method is based on action research principles and can be combined with theoretical validation methods. The industrial experiences reported in their work show than the method is effective in many practical cases.

From all the above examples it can be concluded that TAR is widely used to validate methods in industry. In our work, we employ TAR to validate gestUI in an industrial context to determine its benefits in the field of human-computer interaction related with gesture-based interaction.

IV. EXTENSION OF GESTUI TO SUPPORT FLEXIBILITY

gestUI [9][29] is designed to produce gesture-based user interfaces using multi-stroke and custom gestures based on touch. gestUI employs models and model transformations to generate a modified version of the existing user interface source code. The obtained source code includes gesture-based interaction specified by means of the definition of gesture-action correspondence and the gestures definition. gestUI is implemented on a tool support described in [29]. This tool support is updated considering the flexibility of gestUI described here.

As a result of the limitations found during initial gestUI applications in non-trivial software development projects, we proposed improving its flexibility for use in complex scenarios, such as collaborative gesture-based interaction development.

The flow of the processes to enhance the gestUI flexibility with the aim of defining custom gestures and including gesture-based interaction in a user interface is characterized in a map representation using strategies and intentions. Fig. 1 shows this flow process in which thick lines and sequence numbers represent the route to complete the process to obtain gesture-based user interfaces. The gestUI strategies included in Fig. 1 are described in Table 1. The columns in this table contain the following information: (i) column "Strategy" refers to each strategy included in the aforementioned figure; (ii) column "Description" contains its description. Note that this flow of processes is generic and can be used for building any gesture-based user interface through gestUI.

Fig. 1 shows the process flow (shown as a sequence of numbers next to the intentions in the map representation) including its extension to add the aforementioned flexibility to the definition of custom gestures and the inclusion of gesture-based interactions. These processes can be described as follows:

Intention 1 (Open a local session): End-user opens a local session in gestUI by entering username and password, as a result of this, a User Identification (UID) is assigned to the end-user with the aim of identifying the gestures defined by the end-user;

Intention 2 (Define a gesture): End-user sketches gestures on a touch-based screen using finger or a pen/stylus;

Intention 3 (Include a gesture in the repository): These gestures are included in a repository that contains all definitions of gestures made by each end-user. The name of each gesture contains the UID of the end-user that defined it, that is, the gesture name contains "UID+description_of_gesture", for instance: User1_letterC can be a gesture name defined by User1 representing a gesture with a shape of the letter C;

Intention 4 (Define gesture catalogue (PIM)): End-user can now select gestures from the repository to obtain a platform-independent gesture catalogue;

Intention 5 (Define target platform): A target platform is specified to apply a model transformation;

Intention 6 (Obtain specific gesture catalogue (PSM)): Each end-user selects his/her previously defined gestures that are stored in the platform-independent gesture catalogue with the aim of performing a model-to-text transformation to obtain the platform-specific gesture catalogue;

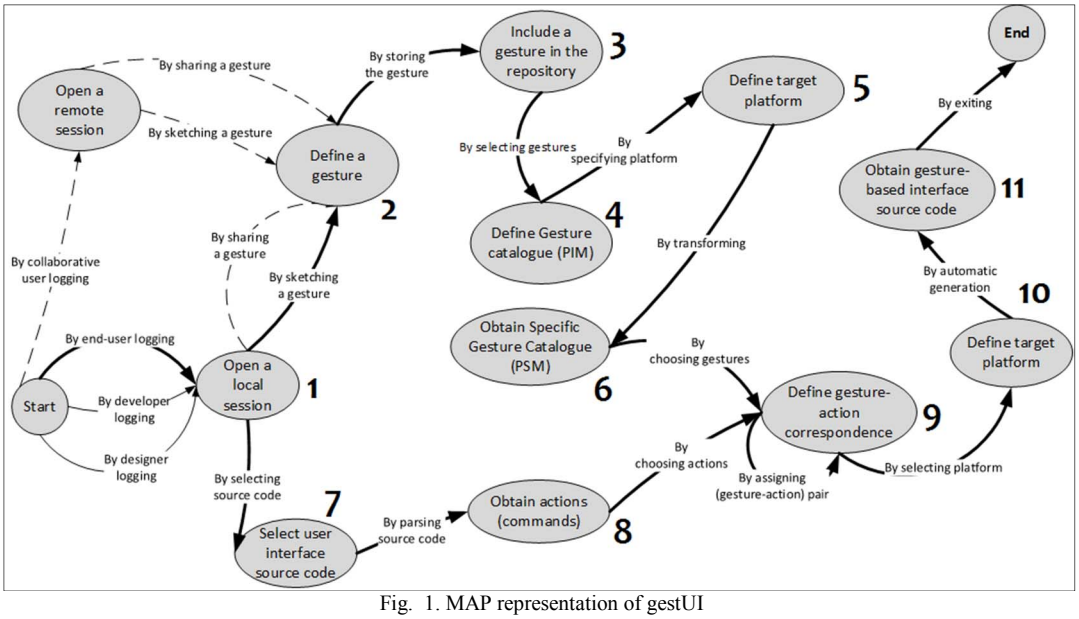


Fig. 1. MAP representation of gestUI

Table 1. Strategies of gestUI

ID	Strategy	Description
1 2 3	Logging in as developer Logging in as end-user Logging in as designer user	Three types of users (developer, designer and end-user) can *log in to gestUI* tool support by means of a local connection using a user name and password. As result of this process, a user identification (UID) is assigned in order to identify the gestures definition of each user to obtain the gesture catalogue definition.
4	Logging in as collaborative user	This type of user can log in to gestUI by means of a remote connection using a username and password. As result of this process, a user identification (UID) is assigned in order to identify the gestures definition of each user to obtain the gesture catalogue definition.
5	By sketching a gestures	Each user can use his fingers or a pen/stylus to *define a single or multi-stroke custom gesture*.
6	By sharing a gesture	Each user can *define a custom gesture* by sharing a gesture which is stored in an external repository.
7	By storing the gesture	Each gesture that is defined by a sketching task or shared from an external repository is *included in the repository* defined by gestUI.
8	By selecting gestures	The user can select gestures from the repository to define *platform-independent gesture catalogue*.
9	By specifying platform	The user *specifies a target platform* with the aim of applying a model transformation.
10	By transforming	Using model transformations a *platform-specific gesture catalogue* is generated.
11	By choosing gestures	The user is required to choose gestures to be considered in the *definition of gesture-action correspondence*.
12	By selecting source code	In order to include gesture-based interaction the *selection of user interface source code* is required.
13	By parsing source code	Applying a parsing process it is possible to *obtain the actions* (commands) included in the user interface source code.
14	By choosing actions	By choosing previously selected actions and gestures the user *defines the gesture-action correspondence* in order to include the gesture-based interaction.
15	By assigning (gesture-action) pair	The user defines the gesture – action relation.
16	By selecting platform	The user *selects the target platform* in order to apply a model transformation to obtain the source code.
17	By automatic generation	A new version of user interface source code is obtained by applying a model-to-text transformation considering previously defined transformation rules and source code generation. The result is the *gesture-based user interface*.
18	By exiting	The process is finished when the user exits gestUI.

Intention 7 (Select user interface source code): The user interface source code is required to determine the actions included in it;

Intention 8 (Obtain actions): The actions included in the source code are obtained through a parsing process;

Intention 9 (Define gesture-action correspondence): End-user defines the gesture-action correspondence (i.e. pairs (gesture, action) in the user interface) to specify the gesture to execute an action;

Intention 10 (Define target platform): The specification of the target platform to apply a model transformation is required.

Intention 11 (Obtain gesture-based interface source code): A new version of the source code of the user interface is obtained through code generation, including gesture-based interaction.

As a result of this process we obtain a user interface supporting gesture-based interaction.

We have updated gestUI tool support including the aforementioned extension. Fig. 2 describes the process using the updated version of gestUI tool support that permits the definition of custom gestures and the inclusion of gesture-based interaction in a user interface.

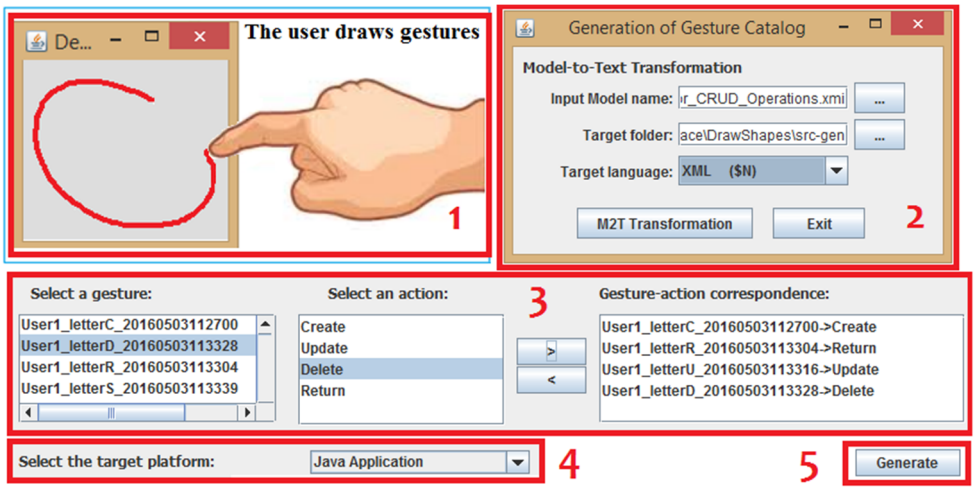


Fig. 2. Including gesture-based interaction by using gestUI tool support

The process starts when the end-user opens a local session in the gestUI tool support (*Intention 1*, not shown in Fig. 2), the other steps of the process, shown in red numbers in Fig. 2, are detailed below. Some intentions are not described in the following paragraphs for reasons of brevity.

Step 1: The tool support captures multi-stroke gestures sketched by each user to transform gestures into a model (*Intention 2*).

Step 2: By using a model transformation, we obtain the platform-specific gesture specification for each user (*Intention 6*).

Step 3: The gesture-action correspondence to specify the action to execute using a previously defined gesture is performed (*Intention 9*).

Step 4: The target platform is selected (*Intention 10*) to apply a model-to-text transformation and to obtain the user interface source code required in the process.

Step 5: Finally, by pressing the "Generate" button we obtain the source code of the user interface source code, including gesture-based interaction (*Intention 11*).

Fig. 2 shows the gesture name that consists of (i) the user name of the software system as mentioned in the description of Intention 3 (e.g. User1), (ii) a name assigned by the user (e.g. letterD), and (iii) the date when the gesture was sketched (e.g. 20160503113328), obtaining the gesture name shown in the aforementioned figure: User1_letterD_20160503113328.

The tool makes use of $N as gesture recogniser to recognize the gestures sketched by the users [35]. In the code generation, gestUI tool includes methods and attributes to add gesture recognition to the software system user interface.

The next step is when end-users use the software system that includes a gesture-based user interface.

The map representation in Fig. 3 describes the process to follow when an end-user uses gestures to perform actions in a user interface supporting gesture-based interaction. Thick lines and sequence numbers represent the route to follow by an end-user to complete the process, employing a software system with gesture-based interaction.

As shown in Fig. 3, when an end-user uses a software with gesture-based interaction generated by gestUI, he/she needs to open a session (*Intention "Log in to IS"*) in the software. In this way, he/she obtains a UID that identifies the gesture catalogue to use in the execution of actions contained in a user interface (*Intention "Use gesture-based user interface"*) by means of custom gestures in the software (*Intention "Use gestures in the user interface"*). In some cases, an end-user could have problems with the custom gestures included in the user interface (e.g. he cannot remember some gesture with a complex definition, or it is hard to sketch). Then he could redefine them and include the new version of the gesture in the gesture catalogue initially included in the user interface (*Intention* "*Redefine custom gestures*" in Fig. 3).

The strategies of the software system included in the gesture-based user interface are described in Table 2. The columns in this table contain the same information as Table 1.

The redefinition of the gesture definition in the software system is performed through the interface shown in Fig. 4 (*Intention* "*Redefine custom gestures*", Fig. 3), and consists of the following steps:

Step 1: End-user selects the option "*Gesture*" in the main menu of the software system (see the main menu of the CDT in Fig. 5). As a result of this selection the user interface to redefine a gesture is shown, containing (Fig. 4): (i) the gesture catalogue defined with gestUI tool support; (ii) a canvas to show the current definition of the selected gesture (Fig. 4, left). By using the "Show it" button, the current definition is shown; (iii) a second canvas to sketch the redefined gesture using a finger or a pen is shown (Fig. 4, right).

Step 2: End-user can select a gesture included in the gesture catalogue, for instance, User2_letterD_20160503113328 in Fig. 4 (*strategy "By selecting gestures"*).

Step 3: If he/she decides to redefine it, then in the second canvas the new gesture definition is sketched (*Strategy "by sketching with a finger"*).

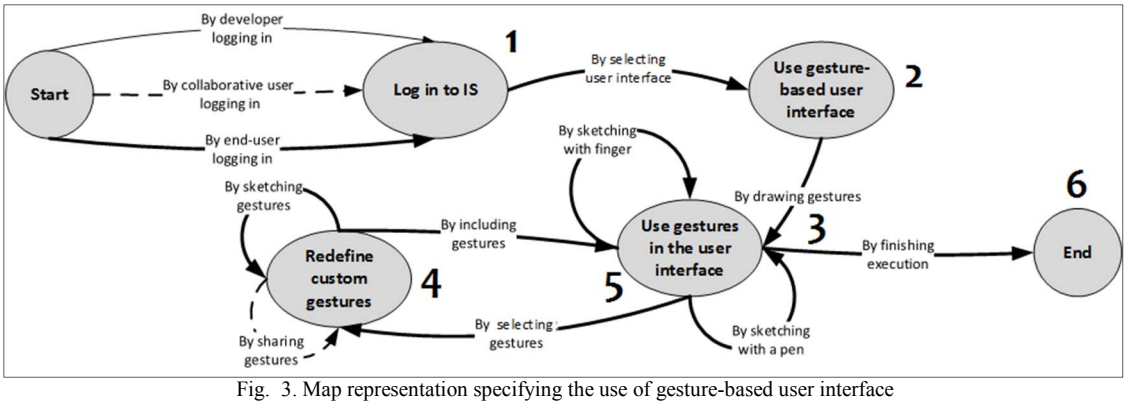


Fig. 3. Map representation specifying the use of gesture-based user interface

Table 2. Strategies defined to use a gesture-based user interface

ID	Strategy	Description
1	Logging in as developer Logging in as collaborative user Logging in as end-user	Three types of users (developer, collaborative and end-user) can *log in to the software system* using a username and password. As result of this process, a user identification (UID) is assigned to identify the definition of the gestures.
2	By selecting user interface	The selection of a *gesture-based UI* is required to use the gesture-based interaction.
3	By drawing gestures	The user sketches previously-defined gestures on the touch-based screen.
4	By redefining gestures	If the user has any problem with the previously defined gesture he can then redefine it.
5	By including gestures	The redefined gestures must be reloaded in the software system to be used in the UI.
6	By finishing execution	The process is finished when the user exits the software system.

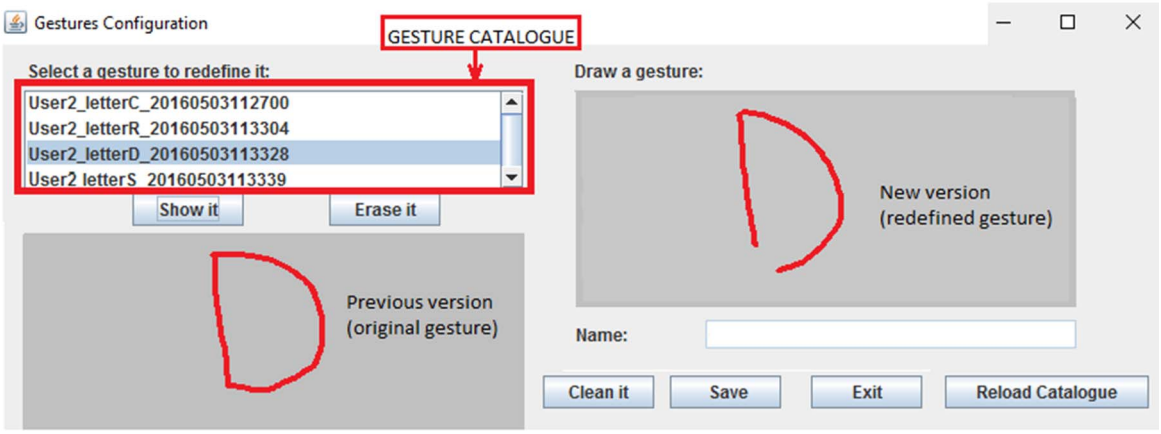


Fig. 4. Interface to redefine a gesture in a software system

Step 4: By using the "Save" button, the new definition of the gesture is included in the gesture catalogue (*Strategy "by including gestures"*).

Step 5: Finally, through the "Reload Catalogue" button, the gesture catalogue containing the redefined gestures is reloaded in the software. Next, the new version of the gestures is available in the software.

V. Capability Design Tool

CDD is a novel paradigm in which services are customised on the basis of the essential business capabilities and delivery is adjusted according to the current context [36]. The CDD methodology for capability-driven design and development consists of various components addressing different modelling aspects, such as context modelling, business services modelling, pattern modelling or capability modelling.

The CaaS project developed three components to support CDD [37]: CDD methodology, Capability delivery patterns, and CDD environment providing a modelling tool called Capability Design Tool (CDT). The CDT is designed as an integrated development environment built using Eclipse Modelling Framework (EMF) technologies (http://www.eclipse.org/emf) and Graphiti (http://www.eclipse.org/graphiti/) on top of Eclipse's Graphical Editing Framework-GEF (http://www.eclipse.org/gef). CDT supports capability modelling according to the CDD metamodel, including context modelling and goal, process and concept models.

We modified the source code of CDT to include gesture-based interaction, which now has two modes of operation: (i) the previous traditional interaction mode, in which the user can manipulate the primitives and connectors contained in a

diagram using mouse and keyboard, and (ii) a new gestural interaction mode, in which the user can draw diagrams by means of gestures sketched by a finger or pen to obtain a primitive.

Fig. 5 shows the CDT interface with gesture-based interaction. In this case, we have included two new elements: (i) the palette (right) allows changing the operation mode between traditional and gesture-based interaction; (ii) the main menu (up) has a new item ("Gesture") to redefine custom gestures based on our approach.

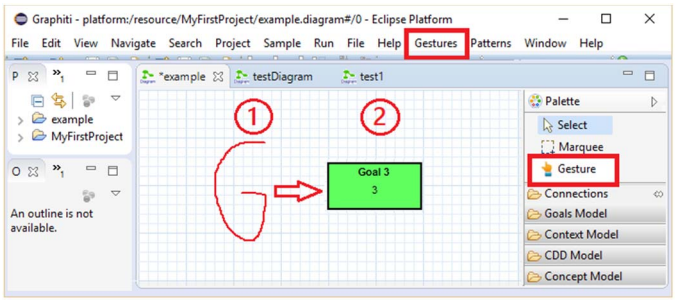


Fig. 5. Gesture-based interface of the CDT

VI. Validation using Technical Action Research

The foundations of this TAR [10] are supported by means of setting up a theoretical framework, which allows the definition of research questions, response variables and their measures.

A. Goal of the TAR.

The goal is to validate gestUI in real-world conditions in relation to its acceptance by means of:

- Perceived Ease of Use (PEOU): this refers to "the degree to which a person believes that using a particular system would be free of effort" [38]. According to Davis [38], when a software system is perceived as easier to use than another, it is more likely to be accepted by users;
- Perceived Usefulness (PU) by the subjects is defined as "the degree to which a person believes that using a particular system would enhance his job performance" [38]. According to Davis [38] if a user perceives the system as an effective way of performing the tasks, then there is a positive user-performance relationship.

In this validation, we wanted to know if gestUI can help software engineers in defining custom gestures and including gesture-based interaction in existing user interfaces.

B. Experimental subjects.

The TAR was conducted in collaboration with two technical analysts from Everis, a partner in the CaaS Project. The technical analysts were women computer engineers with at least 5 years of experience in software development. They also had experience in using CDT with the traditional interaction. They are currently working on a CaaS project using the CDT tool with traditional interaction (keyboard and mouse) and had never seen gestUI before the TAR session.

The background and experience of the subjects were found through a demographic questionnaire handed out at the first session of the experiment. This instrument consists of 15 questions on a 5-point Likert scale.

C. Research questions.

We focused on four research questions:

RQ1: Do the subjects consider that gestUI is easy to use and useful in defining custom gestures?

RQ2: Do the subjects consider that gestUI is easy to use and useful for gesture-based interactions on user interfaces?

RQ3: What is the subject's experience when performing the process of obtaining gesture-based interfaces with gestUI?

RQ4: What is the desirability level of subjects when they use gestUI to generate gesture-based interfaces?

D. Factor and Treatment

In this case, the factor detected in the experiment is the CDT interaction method. This factor has only one treatment: the use of gesture-based interaction. We chose only this treatment since it was the goal of the experiment and the subjects already had knowledge of the process using the traditional interaction (mouse and keyboard).

E. Response variables.

Response variables are the effects studied in the experiment caused by the manipulation of factors. In this experiment, we have four response variables (PEOU, PU, UEQ and MRC) to analyse the acceptance of the Everis technical analysts.

F. Instruments for the TAR.

All the material required to support the experiment was developed beforehand, including the preparation of the experimental object, instruments and task description documents for data collection used during the execution of the experiment. The instruments prepared to perform the TAR are described in Table 3.

G. Experimental Object

With the aim of performing the TAR, we considered CDT as an experimental object in this validation. Using this experimental object, the subjects must sketch an excerpt of a diagram defined in Everis (see Fig. 6), with the primitives included in Table 4. This diagram is an example of work related to a project on the development of an e-government platform.

VII. Action Research Procedure

This section describes the TAR procedure used to conduct the experiment performed in a meeting room in Everis offices. Previous to the TAR session, a pilot test was run with a researcher from the PROS Research Centre in the Universitat Politécnica de Valencia. This pilot test helped us improve the understandability of the instruments.

The steps of the experiment procedure are:

Step 0: The first step is related to the gesture catalogue definition, which was completed for the subjects before the

TAR session. In a previous session, the subjects filled in the Gesture Catalogue Definition Form with the gestures to be used in CDT to draw the aforementioned diagram. The subjects defined custom gestures for each primitive of the aforementioned diagram according to their preferences (Table 4).

Step 1: Before the experiment each subject filled in a Demographic Questionnaire in which they were asked about their experience in tasks related with CDT, experience with gesture-based interaction, experience in software development, and experience in model-driven development.

Step 2: The planned action research procedure was described to the subjects with a verbal explanation.

Table 3. Instruments defined for the validation

Instrument	Description
Gesture Catalogue Definition Form	Form in which the subjects defined the gestures to be used in the TAR to draw diagrams using gestures
Demographic Questionnaire	Questionnaire to assess the subjects' knowledge and experience of the technologies and concepts used in the experiment
Task Description Document	Document that describes the task to be performed in the action research using the gestUI method and containing empty spaces to be filled in by the subjects with start time and end time of the experiment
Post-test Questionnaires	Questionnaire with 16 questions containing Likert-scale values ranging from 1 (strongly disagree) to 7 (strongly agree) to evaluate gestUI performance in the definition of custom gestures and in the inclusion of gesture-based interaction
Product reaction card method	Questionnaire containing 118 positive and negative adjectives, employed by the subjects to describe their experience *[16]* with gestUI.
User Experience Questionnaire (UEQ)	Questionnaire to obtain fast and immediate measurement of user experience of interactive products. This questionnaire contains 6 scales with a total of 26 items: Attractiveness (Annoying/enjoyable, unlikable/pleasing, unpleasant/pleasant, good/bad, attractive/unattractive, and friendly/unfriendly); Efficiency **(**Fast/slow, inefficient/efficient, impractical/practical, and organized/cluttered**);** Perspicuity **(**Not understandable/understandable, easy to learn/difficult to learn, complicated/easy, clear/confusing**);** Dependability **(**Unpredictable/predictable, obstructive/supportive, secure/not secure, meets expectations/does not meet expectations**);** Stimulation **(**Valuable/inferior, boring/exciting, not interesting/interesting, and motivating/demotivating**);** Novelty **(**Creative/dull, inventive/conventional, usual/leading edge, conservative/innovative**)**

Step 3: By means of a live demo, the subjects were instructed to use gestUI in gesture definition and inclusion of gesture-based interaction on the user interface of CDT.

Step 4: Subjects used gestUI to define the gestures previously specified in the Gesture Catalogue Definition Form (Table 4) following the process defined in Section IV specified in the *intention* "*Define a gesture*" in Fig. 1. Subjects used the Task Description Document to follow the required instructions in order to obtain the gesture catalogue (Intention "*Obtain specific gesture catalogue (PSM)*" in Fig. 1), and to include gesture-based interaction in the interface of CDT (Intention "*Obtain gesture-based interface source code*" in Fig. 1).

Step 5: Subjects filled in the Post-Task Questionnaire on their opinion of gestUI regarding custom gesture definition and inclusion of gesture-based interaction in CDT.

Step 6: Subjects employed CDT to draw the diagram shown in Fig. 6. They used the Gesture Catalogue Definition Form to help them with the previously defined gestures (*Intention* "*Use gestures in the user interface*" in Fig. 3).

Step 7: Subjects redefined three gestures using the module to redefinition included in CDT (*Intention* "*Redefine custom gestures*" in Fig. 3).

Step 8: Subjects filled in the Post-Task Questionnaire to assess gestUI capacity to define custom gestures and to include gesture-based interaction.

Step 9: Subjects filled in the User Experience Questionnaire on their experience with custom gesture definition and the inclusion of gesture-based interaction.

Step 10: Subjects filled in the Microsoft Reaction Cards on the desirability level of using gestUI to define custom gestures and include gesture-based interaction.

Table 5 contains a summary of the steps performed in the experiment, the instruments used in each step and the time estimated to perform each step.

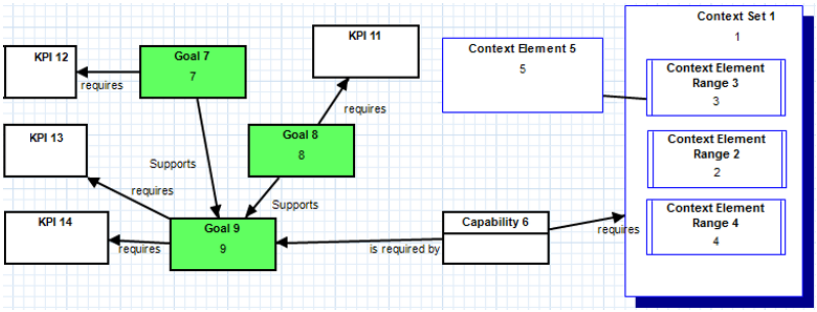


Fig. 6. Excerpt of a model defined in Everis

Table 4. Gesture catalogue defined by the subjects

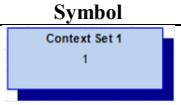
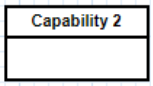
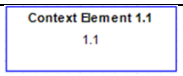
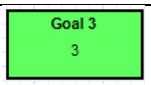
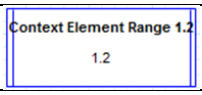
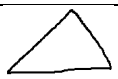
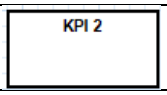
Primitive	Symbol	Gesture	Primitive	Symbol	Gesture
Context Set			Capability		
Context Element			Goal		
Context Element Range			KPI		

Table 5. A summary of the experiment procedure

ID	Description	Instrument used	Time
1	Subjects filled in the *Demographic Questionnaire* on their experience in related topics.	Demographic questionnaire	8 min.
2	The planned action research procedure was described to the subjects.	Verbal explanation	10 min.
3	Subjects were instructed to use gestUI in gesture definition and inclusion of gesture-based interaction on the user interface of the CDT.	Live demo	10 min.
4	Subjects employed gestUI to define the gestures previously specified in the gesture catalogue. They employed the *Task Description* document to follow the required instructions.	Task Description Document	15 min.
5	Subjects filled in the Post-Task Questionnaire on their opinion of gestUI.	Post-Task Questionnaire	5 min.
6	Subjects employed the CDT to draw the diagram shown in Fig. 6. They used the *Gesture Catalogue Definition Form* to help them with the previously defined gestures.	Task Description Document and Gesture Catalogue Definition Form	20 min.
7	Subjects redefined three gestures using the module to redefinition included in the CDT	Task Description Document	10 min.
8	Subjects filled in the PEOU and PU Post-Task Questionnaire to assess gestUI capacity to define custom gestures and include gesture-based interaction.	Post-Task Questionnaire	5 min.
9	Subjects filled in the User Experience Questionnaire on their experience with custom gesture definition and the inclusion of gesture-based interaction.	User Experience Questionnaire	5 min.
10	Subjects filled in the reaction cards on the desirability level of using gestUI to define custom gestures and include gesture-based interaction.	Microsoft Reaction Cards	8 min.
		Total time	**96 min.**

Table 6. Results obtained from the UEQ

Scale	Description	Value obtained	
		Custom gesture definition	Gesture-based interaction
Attractiveness	Overall impression of the product	2.42 (81%)	2.25 (75%)
Perspicuity	Is it easy to get familiar with the product?	2.75 (92%)	2.75 (92%)
Efficiency	Can users solve their tasks without unnecessary effort?	1.38 (46%)	1.63 (54%)
Dependability	Does the user feel in control of the interaction?	1.63 (54%)	2.00 (67%)
Stimulation	Is it exciting and motivating to use the product?	2.25 (75%)	2.50 (83%)
Novelty	Is the product innovative and creative?	2.38 (79%)	2.63 (88%)

VIII. Analysis and Interpretation of Results

As there were only 2 subjects involved in the TAR we did not apply any statistical test to analyse and interpret the information. We analysed the responses of each subject regarding each research question obtained from the aforementioned instruments containing the questionnaires filled in by the subjects:

Regarding RQ1, the results obtained through the questionnaires show that both subjects think that the feature to define custom gestures implemented in gestUI is perceived as both easy to use and useful.

Regarding RQ2, the results obtained from the questionnaires show that both subjects think that the feature to include gesture-based interaction implemented in gestUI is perceived as both easy to use and useful.

Regarding RQ3, after completing the tasks, the subjects filled out the UEQ, obtaining the results shown in Fig. 7 and Fig. 8. The values vary from -3 to +3. The six scales, their description [14], the values obtained and their corresponding percentages in the TAR are shown in Table 6. The results obtained show that efficiency and dependability scales in custom gesture definition had values lower than 67%. The efficiency scale in gesture-based interaction also had a value lower than 67%. In both cases, efficiency is related to items such as: fast/slow, inefficient/efficient, impractical/practical, and organized/cluttered.

Regarding RQ4, we applied MRC to study positive and negative aspects related with the inclusion of gesture-based interaction (blue line) and custom gesture definition (orange line). With the aim of reporting these results, we use two figures: (i) Fig. 9 shows positive results and Fig. 10 shows negative results. Values showed in Fig. 9 represent the

frequency of use of each positive adjective for the subjects in the experiment (e. g. “simplistic” was selected two times, one time per subject). These values correspond to the values included in the "Value" column in Table 7, which shows the most frequently used positive adjectives on the gestUI experience.

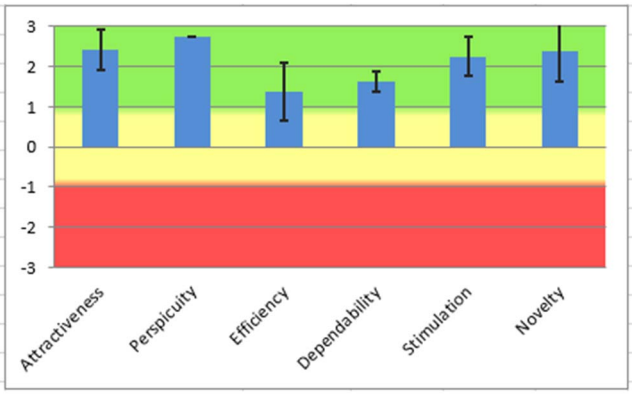


Fig. 7. UEQ results: custom gesture definition interaction

From Fig. 10 we obtained the negative results related with the inclusion of gesture-based interaction and custom gesture definition shown in Table 8. The meaning of the values included in this figure is the same as in Fig. 9.

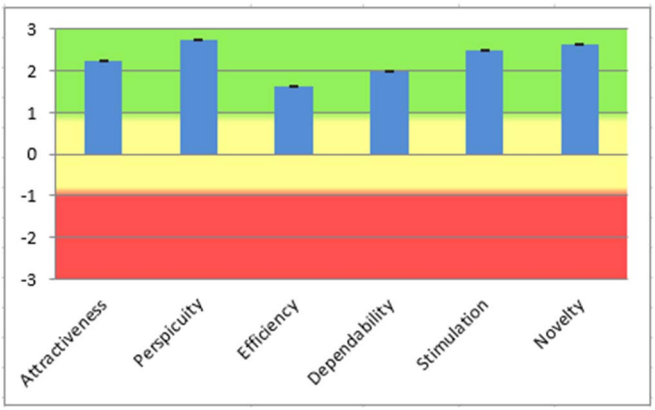


Fig. 8. UEQ results: inclusion of gesture-based interaction

Table 7. Reaction cards positive results

Process	Positive Adjective	Value
Custom Gesture Definition	Simplistic	2
	Innovative, Customizable, Useful, Clear, Easy to use.	1
Inclusion of Gesture-based Interaction	Innovative, Useful	2
	Comfortable, Creative, Attractive, Time saving, Simplistic, Easy to use.	1

Table 8. Reaction cards negative results

Process	Negative Adjective	Value
Custom Gesture Definition	Too technical	2
	Time consuming, Unattractive	1
Inclusion of Gesture-based Interaction	Slow	2
	Sensible, Annoying, Fragile	1

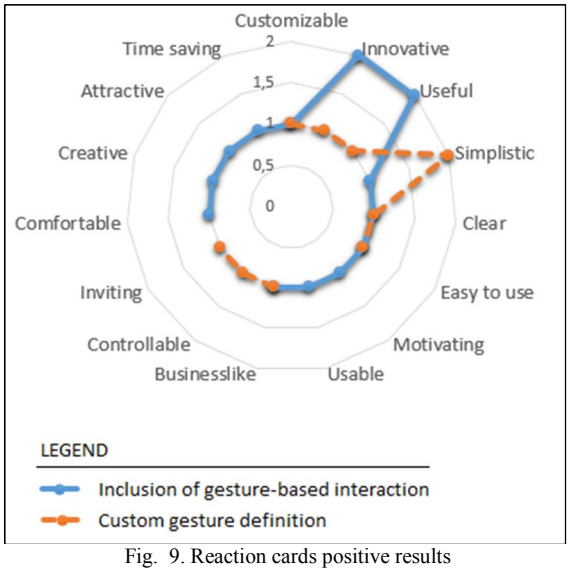


Fig. 9. Reaction cards positive results

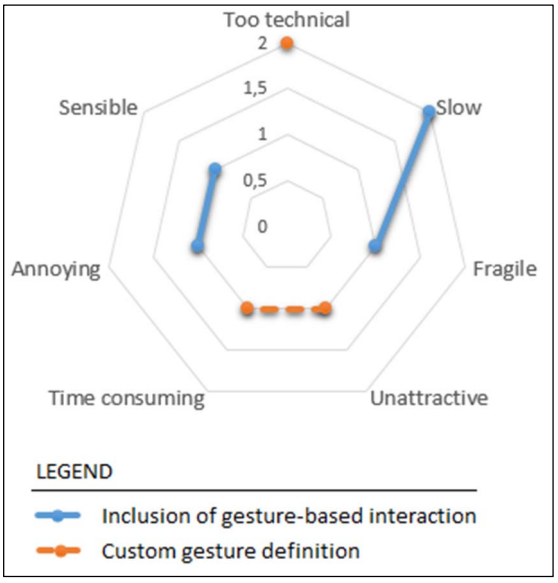


Fig. 10. Reaction cards negative results

In the case of custom gesture definition, the subjects described the custom gesture definition as simplistic but also too technical and time consuming. This opinion could have been related with the null experience of the subjects in custom gesture definition in using CDT and also because the UI of gestUI to define gestures could have been better designed to obtain an attractive gesture definition process. In the case of the inclusion of gesture-based interaction, the subjects defined it as innovative and useful, but also that the inclusion of gesture-based interaction is slow, sensitive and annoying. This

opinion could have been due to the null experience of the subjects in the use of gestures to draw diagrams.

The subjects considered the new proposal to redefine gestures as useful and thought that it helped them to solve memorizing or sketching problems.

IX. Threats to Validity

This section deals with the most important threats to the validity of this evaluation, classified according to Wohlin et al. [39]:

(A) Internal Validity: the main threats to the internal validity of the experiment are: (1) *Subject experience in tasks performed in the experiment*: this threat was eliminated since none of the subjects had any experience in tasks related with custom gestures definition or the inclusion of gesture-based interaction in user interfaces. (2) *Subject experience in the use of CDT with gesture-based interaction*: this threat was eliminated since none of the subjects had any experience in the use of CDT with gesture-based interaction.

(B) External Validity: the main threat to the external validity of the experiment was: (1) *Duration of the experiment*: since the duration of the experiment was limited to 96 minutes, only one diagram was selected with six primitives and six gestures. However, experience in the use of CDT in traditional interaction and repetitive tasks could have affected the duration of the experiment, since the subjects already knew the process to be performed. This threat could not be ruled out since they were familiar with the repetitive tasks required to build the diagram. (2) *Representativeness of the results*: the experiment was performed in an industrial context on subjects with no experience in the tasks related with the experiment. This means the results could only be representative for novice evaluators with no experience in custom gesture definition and in the inclusion of gesture-based interaction.

(C) Validity Conclusions: The main threat to the validity of the experiment was: (1) *Validity of the statistical test applied*: In this case, we did not apply any statistical tests to obtain answers to the research questions because the sample size was too small. However, we did consider the results obtained with other methods, such as MRC and UEQ.

X. Conclusions and Future Work

This paper describes the enhancement of the model-driven gestIU method aimed at increasing its flexibility for developing gesture-based software interfaces. The paper also contributes with the validation of gestUI in industry by means of a TAR, studying (i) PEOU; (ii) PU; (iii) the desirability level with MRC; and, (iv) user experience with UEQ.

The enhancement of gestUI consists of a new engineering framework that includes MAP diagrams and additional method components (e.g. Strategies 4 and 6 in Table 1) to enable flexibility in the interaction (e.g. collaboration). To validate the performance of gestUI in industrial settings, we included gesture-based interaction in the CDT tool from the CaaS Project. The subjects were two business analysts from a consultancy firm who defined custom gestures by either fingers or pen/stylus and also redefined some gestures from the gesture catalogue considered in the experiment.

The main findings of the study are: (1) gestUI helped the business analysts to define custom gestures and include gesture-based interaction in user interfaces. (2) The subjects considered gestUI easy to use and useful for defining custom gestures and including gesture-based interaction in CDT. (3) Although the subjects did not enjoy defining custom gestures and applying the automated transformations, they did feel motivated while using this version of the CDT.

In future work we plan to include new automated transformations that target new platforms (e.g. mobile devices). Further validation will be undertaken in other development scenarios to evaluate collaborative features, as well as to generalize the results and compare different types of developers and users.

Further details of gestUI and the validation described in this paper can be found at https://gestui.wordpress.com/tar/.

Acknowledgments

This work has been supported by the Universidad de Cuenca and SENESCYT of Ecuador, and received financial support from the Generalitat Valenciana under Project IDEO (PROMETEOII/2014/039) and the Spanish Ministry of Science and Innovation through the DataMe Project (TIN2016-80811-P).

References

[1] F. Karray, M. Alemzadeh, J. A. Saleh, and M. N. Arab, "Human-Computer Interaction : Overview on State of the Art," *Int. J. Smart Sens. Intell. Syst.*, vol. 1, no. 1, pp. 137–159, 2008.

[2] X. Yan and N. Aimaiti, "Gesture-based interaction and implication for the future," 2011.

[3] B. Poppinga, A. Sahami Shirazi, N. Henze, W. Heuten, and S. Boll, "Understanding shortcut gestures on mobile touch devices," *Proc. 16th Int. Conf. Human-computer Interact. with Mob. devices Serv. - MobileHCI '14*, pp. 173–182, 2014.

[4] M. Hesenius, T. Griebe, S. Gries, and V. Gruhn, "Automating UI tests for mobile applications with formal gesture descriptions," *MobileHCI 2014 - Proc. 16th ACM Int. Conf. Human-Computer Interact. with Mob. Devices Serv.*, pp. 213–222, 2014.

[5] P. Parviainen, J. Takalo, S. Teppola, and M. Tihinen, *Model-Driven Development Processes and practices*. Finland: Julkaisija Utgivare Publisher, 2009.

[6] O. Pastor and J. Molina, *Model-Driven Architecture in practice: a software production environment based on conceptual modeling*. Springer-Verlag Berlin Heidelberg, 2007.

[7] P. E. Papotti, A. F. Do Prado, W. L. De Souza, C. E. Cirilo, and L. F. Pires, "A quantitative analysis of model-driven code generation through software experimentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7908 LNCS, pp. 321–337, 2013.

[8] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, MA: Addison-Wesley Longman Publishing, Inc., 2003.

[9] O. Parra, S. España, and O. Pastor, "gestUI : A Model-driven Method and a Tool for Developing Gesture-based Information System Interfaces," in *Proceedings of the CAiSE'15 Forum at the 27th International Conference on Advanced Information Systems Engineering*, 2015, vol. Stockholm, pp. 49–56.

[10] R. Wieringa and A. Moralı, "Technical Action Research as a Validation Method in Information Systems Design Science," *Des. Sci. Res. Inf. Syst. Adv. Theory Pract.*, vol. 7286, pp. 220–238, 2012.

[11] S. Bērziša, G. Bravos, T. C. Gonzalez, U. Czubayko, S. España, J.

Grabis, M. Henkel, L. Jokste, J. Kampars, H. Koç, J.-C. Kuhr, C. Llorca, P. Loucopoulos, R. J. Pascual, O. Pastor, K. Sandkuhl, H. Simic, J. Stirna, F. G. Valverde, and J. Zdravkovic, "Capability Driven Development: An Approach to Designing Digital Enterprises," *Bus. Inf. Syst. Eng.*, vol. 57, no. 1, pp. 15–25, 2015.

[12] M. España, S.; Grabis, J.; Henkel, "Strategies for Capability Modelling: Analysis Based on Initial Experiences," *Lect. Notes Bus. Inf. Process.*, vol. 112, pp. 123–129, 2012.

[13] C. Rolland, "Capturing System Intentionality with Maps," in *Conceptual Modelling in Information Systems Engineering*, J. Krogstie, A. L. and Opdahl, and S. and Brinkkemper, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 141–158.

[14] D. L. Moody, "The Method Evaluation Model : A Theoretical Model for Validating Information Systems Design Methods," *Inf. Syst. J.*, pp. 1327–1336, 2003.

[15] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Applying the user experience questionnaire (UEQ) in different evaluation scenarios," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8517 LNCS, no. PART 1, pp. 383–392, 2014.

[16] T. Merčun, "Evaluation of information visualization techniques: analysing user experience with reaction cards," *Proc. Fifth Work. Beyond Time Errors Nov. Eval. Methods Vis.*, pp. 103–109, 2014.

[17] P. Soffer and C. Rolland, "Combining Intention-Oriented and State-Based Process Modeling," in *Proceedings Conceptual Modeling - ER2005*, 2005, pp. 47–62.

[18] C. M. Barnum and L. A. Palmer, "Tapping into Desirability in User Experience," in *Usability of Complex Information Systems: Evaluation of User Interaction*, M. Albers and B. Still, Eds. CRC Press, 2011, pp. 253–279.

[19] S. Adikari, C. McDonald, and J. Campbell, "Quantitative Analysis of Desirability in User Experience," *Australasian Conference on Information Systems*, 2015. [Online]. Available: https://acis2015.unisa.edu.au/wp-content/uploads/2015/11/ACIS_2015_paper_230.pdf. [Accessed: 23-Jan-2017].

[20] H. Santoso, M. Schrepp, R. Kartono Isal, A. Yudha, and B. Priyogi, "Measuring User Experience of the Student-Centered e-Learning Environment," *J. Educ. On-line*, vol. 13, no. 1, pp. 58–79, 2016.

[21] M. Rauschenberger, M. Schrepp, M. Perez-Cota, S. Olschner, and J. Thomaschewski, "Efficient Measurement of the User Experience of Interactive Products. How to use the User Experience Questionnaire (UEQ).Example: Spanish Language Version," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 2, no. 1, pp. 39–45, 2013.

[22] A. Nawaz, J. L. Helbostad, L. Chiari, F. Chesani, and L. Cattelani, "User Experience (UX) of the Fall Risk Assessment Tool (FRTA-up)," in *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, 2015, pp. 19–22.

[23] B. Laugwitz, T. Held, and M. Schrepp, "Construction and Evaluation of a User Experience Questionnaire," *HCI Usability Educ. Work*, pp. 63–76, 2008.

[24] F. Beuvens and J. Vanderdonckt, "Designing graphical user interfaces integrating gestures," *Proc. 30th ACM Int. Conf. Des. Commun. - SIGDOC'12*, pp. 313–322, 2012.

[25] M. de Paiva Guimaraes, V. F. Martins, and J. R. F. Brega, "A software development process model for gesture-based interface," *Syst. Man, Cybern., 2012 IEEE Int. Conf.*, pp. 2985–2990, 2012.

[26] M. Nielsen, M. Störring, T. B. Moeslund, and E. Granum, "A PROCEDURE FOR DEVELOPING INTUITIVE AND ERGONOMIC GESTURE INTERFACES FOR MAN-MACHINE INTERACTION Michael Nielsen , Moritz Störring , Thomas B . Moeslund , and Erik Granum Aalborg University , Laboratory of Computer Vision and Media Technology , Niels Jer," *Technology*, pp. 1–12, 2003.

[27] A. Bragdon, R. Zeleznik, B. Williamson, T. Miller, and J. J. LaViola, "GestureBar: Improving the Approachability of Gesture-based Interfaces," *Proc. 27th Int. Conf. Hum. factors Comput. Syst. - CHI 09*, pp. 2269–2278, 2009.

[28] M. Bhuiyan and R. Picking, "A Gesture Controlled User Interface for Inclusive Design and Evaluative Study of Its Usability," *J. Softw. Eng. Appl.*, vol. 4, no. September, pp. 513–521, 2011.

[29] O. Parra, S. España, and O. Pastor, "GestUI : A Model-driven Method and Tool for Including Gesture-based Interaction in User Interfaces," *Complex Syst. Informatics Model. Q.*, no. 6, pp. 73–92, 2016.

[30] O. Parra, S. España, and O. Pastor, "Including multi-stroke gesture-based interaction in user interfaces using a model-driven method," in *Proceedings of the XVI International Conference on Human Computer Interaction - Interacción '15*, 2015, pp. 1–8.

[31] M. Morales-Trujillo, H. Oktaba, and M. Piattini, "Using Technical-Action-Research to Validate a Framework for Authoring Software Engineering Methods," *Proc. 17th Int. Conf. Enterp. Inf. Syst.*, pp. 15–27, 2015.

[32] A. Morali and R. Wieringa, "Risk-based confidentiality requirements specification for outsourced IT systems," *Proc. 2010 18th IEEE Int. Requir. Eng. Conf. RE2010*, pp. 199–208, 2010.

[33] U. Abelein, "User-Developer Communication in Large Scale IT Projects," Heidelberg University, 2015.

[34] V. Antinyan, M. Staron, and A. Sandberg, "Validating Software Measures Using Action Research A Method and Industrial Experiences," in *Proceedings of the 17th International Conference on Enterprise Information Systems - (Volume 2)*, 2016, pp. 15–27.

[35] L. Anthony and J. O. Wobbrock, "A lightweight multistroke recognizer for user interface prototypes," *Proc. Graph. Interface 2010*, pp. 245–252, 2010.

[36] J. Stirna, J. Grabis, M. Henkel, and J. Zdravkovic, "LNBIP 134 - Capability Driven Development – An Approach to Support Evolving Organizations," pp. 117–131, 2012.

[37] S. España, T. Gonzalez, J. Grabis, L. Jokste, R. Juanes, and F. Valverde, "Capability-driven development of a SOA platform: A case study," *Lect. Notes Bus. Inf. Process.*, vol. 178 LNBIP, pp. 100–111, 2014.

[38] F. D. Davis, "Perceived Usefulness , Perceived Ease Of Use , And User Acceptance," *MIS Q.*, vol. 13, no. 3, pp. 319–339, 1989.

[39] C. Wohlin, M. Host, P. Runeson, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering: an introduction*. Springer Berlin Heidelberg, 2012.