Including Gesture-based Interaction in Capability Design Tool

Otto Parra-González[1, 2], Sergio España[1], Oscar Pastor[1]

[1] Universitat Politècnica de València, Cami de Vera, 46022, Valencia, Spain

[2] Universidad de Cuenca, Av. 12 de Abril y A. Cueva, Cuenca, Ecuador

otpargon@upv.es, sergio.espana@dsic.upv.es, opastor@dsic.upv.es

Abstract. Currently, the process of definition of custom gestures and the inclusion of gesture-based interaction in user interfaces of information systems is complicated because to the diversity of devices, software platforms and development tools available. In order to help to resolve this situation we had proposed gestUI, a model-driven method, whose objectives are custom gesture definition and gesture-based user interface source code generation. This paper describes gestUI using the "Template for the Documentation of Method Components in CaaS" and it demonstrates the application of gestUI in order to include gesture-based interaction in the tool support called Capability Design Tool (CDT) of CaaS project.

Keywords. Gesture-based interaction, capability, information systems, user interaction, model-driven method

1 Introduction

Currently, there are some types of devices that the users employ (e.g., mobile phone, tablet, notebook), in conjunction with this devices, there are some types of interaction (e.g., gesture-based, gaze-based, voice-based interaction) [1] that the users have available. One of the more widespread types of interaction is gesture-based interaction.

Interfaces supporting gesture-based interaction have been reported to be more challenging to implement and test than traditional mouse and pointer interfaces [2]. Gesture-based interaction is supported at the source code level (typically third-generation languages) [3]. This comprises a great coding and maintenance effort when multiple platforms are targeted, has a negative impact on reusability and portability, and complicates the definition of new gestures. Some of these challenges can be tackled by following a Model-driven Development (MDD) approach provided that gestures and gesture-based interaction can be modelled and that it is possible to automatically generate the software components that support them.

We had proposed gestUI [4], a model-driven method for gesture-based information systems user interface development, which is intended to allow software engineers to focus on the key aspects of gesture-based information system interfaces; namely, defining customized gestures and specifying gesture-based interaction.

Capability is the ability and capacity that enable an enterprise to achieve a business goal in a certain context [5]. Capability-driven development (CDD) is a novel paradigm where services are customised on the basis of the essential business capabilities and delivery is adjusted according to the current context [6]. The CDD methodology [7] for capability-driven design and development will consist of various components addressing different modelling aspects, such as context modelling, business services modelling, pattern modelling or capability modelling [7]. This methodology considers the definition of method components. A method component should consists of concepts, a procedure and a notation. Each procedure has input, output, and tool support [8].

In order to support CDD, the CaaS project has planned the following components [9]:

- CDD methodology: an agile methodology for identification, design and delivery of context aware business models. The notion of capability is formalised by means of a metamodel containing the following elements: goal, key performance indicator (KPI), context, capacity and, ability.
- Capability delivery patterns: they are generic organisational design and business process which can be easily adapted, reused, and executed.
- CDD environment: tool support for design and runtime of CDD solutions. Its first release is called Capability Design Tool (CDT), it is designed as an integrated development environment built on the Eclipse Framework, using the Eclipse Modelling Framework (EMF) technologies. This tool supports capability modelling according to the capability metamodel including context modelling and goal, process and concepts models

The contributions of this paper are: (i) the description of the method gestUI by means of the "Template for the Documentation of Method Components in CaaS" and, (ii) the demonstration of use of gestUI with the aim to include gesture-based interaction in a CASE tool, in this case, Capability Design Tool (CDT).

The remainder of this paper is organized as follows. Section 2 contains the state of the art. Section 3 describes gestUI. Section 4 includes a description of Capability Design Tool (CDT). Section 5 contains details of the application of the method gestUI in a CASE tool, in this case CDT of CaaS project. The paper ends with some conclusions and future work in Section 6.

2 State of the Art

This section presents a related literature review of gesture-based human-computer interaction.

Vanderdonckt [10] describes in his work a set of variables associated to the development of user interfaces, one of which contemplates interaction devices and styles. Interaction styles include the gesture recognition. He points out that an abstract user interface is independent of any interaction modality [11] so that an explicit reference to a specific type of interaction style is not considered.

The authors of [12] include a report related with user interface plasticity and MDE, in which three information spaces are defined: the user model, environment model, and platform model. The platform model considers the possible interactions that can be included in a user interface. This report also includes a description of models that have been defined with the aim of creating user interfaces. It also mentions the variety of interaction modalities currently available thanks to the diversity of technological spaces that can be included in the definition of concrete user interfaces.

Calvary et al. in [13] describe the relation between MDE and HCI in implementing user interfaces. In this context, they introduce the models contained in a number of frameworks (e g., UsiXML, CTTe), one being the interaction model considered in the process of defining user interfaces. However, the interaction modality is not considered.

Carvalho et al. describe in their work [14] the gesture-based interaction in domotic environments considering three dimensions: people, gesture mode of interaction and domotics. They grouped the gestural interaction as perceptual and non-perceptual and they describe some types of applications in the field of domotics that employs these two types of interaction. In order to identify and discuss the HCI-related aspects and challenges of multimodal interaction, they propose a socio-technical framework of multimodal interaction in the context of intelligent domotics. The framework consists of the main dimensions technology, modes of interaction, and people.

Our proposal considers a model-driven method to define custom multi-strokes gestures using the fingers of the user, it also consist of a process to include gesture-based interaction in user interfaces of information systems.

3 gestUI

3.1 Describing gestUI

gestUI [4] is a user-driven and iterative method that follows the MDD paradigm [15]. The main artefacts are models that conform to the Model-Driven Architecture [16], a generic framework of modelling layers that ranges from abstract specifications to the

software code. gestUI permits to define multi-strokes gestures, obtaining the gesture catalogue model and, using model transformations to obtain the source code of interfaces including gesture-based interaction (Fig. 1).

Applying model-driven development method the results obtained with gestUI are:

- Model-driven gesture catalogue which contains the definition of gestures using XML. This definition is platform-independent and it can be imported in frameworks that employ gestures as input in their process.
- Source code to include in the specification of user interface in order to include gesture-based interaction in the user interface of information systems. The method produces source code in the selected technology containing the definition of gesture-based interaction.

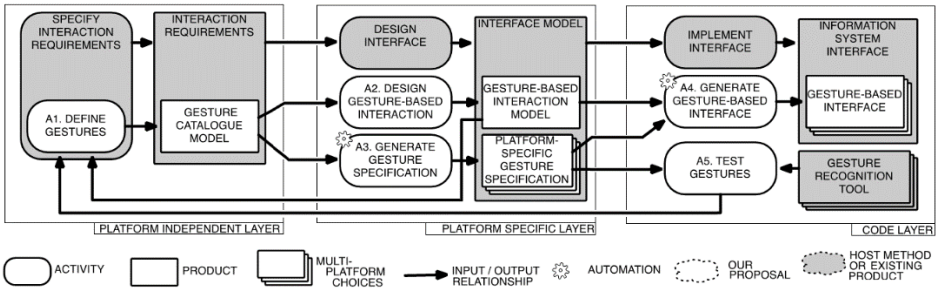

Fig. 1. Method proposed

3.2 Method components

Considering the "Template for the Documentation of Method Components in CaaS" we describe gestUI by means of following method components:

a. Method component to define customised gestures: it permits the definition of customised gestures in order to obtain a platform-independent gesture catalogue to include in the specification of the user interaction of an information system.
b. Method component to generate platform-specific gesture specification: it permits stakeholders to obtain gestures specification according to the platform specified by them. The process includes a model-to-text transformation with the aim of produce a platform-specific gesture catalogue.
c. Method component to design gesture-based interaction: it consists of the specification of the relation between gestures and actions contained in an information system. A parsing process is executed on the user interface source code to determine the actions in such user interface that will be associated with previously defined gestures.

d. Method component to generate gesture-based interface. It is added source code in the user interface source code containing the gesture-based interaction.
e. Method component to test gestures. This component permits to test the defined previously gestures. If the user is not according with the definition of a gesture, it is possible redefine the gesture.

The method components have an execution sequence that permits obtain some products: gesture-catalogue model, gesture-based interaction model, platform-specific gesture specification, and gesture-based interface. Each product that is obtained in a method component is the input to other method component, as shown in Fig. 2.

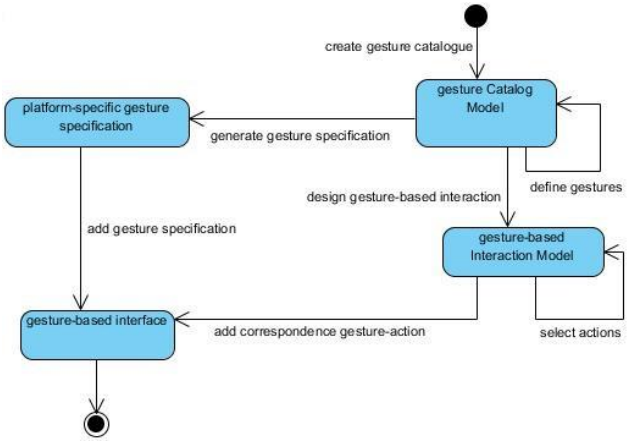

Fig. 2. Method components

The method component "*Definition of gestures*" contains a process to define gestures in order to conform a gesture catalogue model. The information of the procedure of this method component is: (1) **Input**: set of strokes defined by coordinates (X, Y) and a timestamp; (2) **Output**: a platform-independent gestures catalogue; (3) **Tool support**: a user interface where the user draws a gesture.

The procedure of this method component comprises of different work steps to be performed:

- The user sketches one or more strokes using his/her finger on the screen. These strokes are stored in a data structure, conforming a gesture catalogue model.
- If the gesture is according to the requirements of the user/system, then the gesture is saved.
- The set of gestures that have been sketched by the user defines the gesture catalogue model, which is conforms to gesture metamodel (see Fig. 3).

The next method component is called “Generation of platform-specific gesture specification”. With this method component the platform-specific gesture specification is obtained by means of a model transformation. The information of the procedure of this method component is: (1) **Input**: the platform-independent gesture catalogue, the target platform and target folder; (2) **Output**: the platform-specific gesture specification; (3) **Tool support**: a model transformation.

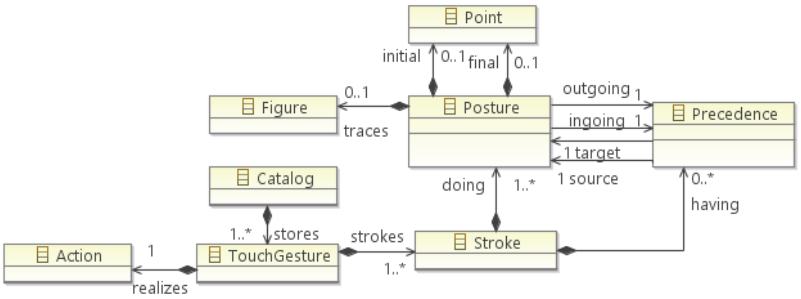

Fig. 3. Gesture catalogue metamodel

The different work steps to be performed in the procedure are:

- The user specifies a set of gestures that will conform the platform-specific gesture catalogue. The gestures are selected from a repository containing gestures previously defined.
- Then, the user specifies the target platform in order to apply transformation rules to obtain the specific solution.
- Finally, the user specifies the target folder where the gesture catalogue will be stored.

Other method component is called “*Design of gesture-based interaction*”. In order to obtain this design, gestUI requires the specification of a user interface where actions (commands) will be executed using gestures. This method component applies a parsing process on the user interface source code with the aim of analyse it to obtain actions included in this code (e g., actions defined using structure “action perform” in Java). Then, the user defines the correspondence between gestures and actions selecting the gestures contained in the catalogue and specifying the action to execute by means of this gesture.

The information of the procedure of this method component is: (1) **Input**: user interface source code; (2) **Output**: the definition of the correspondence gesture-action; (3) **Tool support**: a user interface where the user specify the input to this procedure.

The different work steps to be performed in this method component are:

- The user specifies a user interface source code, for instance, source code in Java containing “action listener” which specify actions to execute.
- Parsing the source code in order to find actions included, e.g., actions defined in the “action listener” contained in the source code.
- The user define a correspondence between actions and gestures previously selected from the gesture catalogue model.

The method component called “*Generation of gesture-based interface*” permits the generation of source code of user interface considering gesture-based interaction. The information of the procedure of this method component is: (1) **Input**: user interface source code, correspondence gesture-action; (2) **Output**: user interface source code containing gesture-based interaction; (3) **Tool support**: a user interface where the user specify the input to this procedure.

The different work steps to be performed in this method component are:

- The user interface source code is specified in order to include gesture-based interaction specification.
- The correspondence gesture-action is specified.
- A process is executed in order to analyse the source code and to insert the source code related with the actions and the gestures defined previously. The output of this process is the source code of the user interface containing the gesture-based interaction.

The last method component, called “*Testing gestures*”, permits that the users test the gestures defined with this method by means of an existent framework (e. g., iGesture, $N, quill). In this case, as first step, the gesture catalogue, previously defined using gestUI, is imported in the framework, then in the next step, the user employs the functionalities available in the framework in order to test the gestures.

The information of the procedure of this method component is: (1) **Input**: gesture catalogue obtained by means of a model transformation; (2) **Output**: information about the gesture testing process; (3) **Tool support**: an existent framework to test the gesture catalogue.

The different work steps to be performed in this method component are:

- A gesture catalogue defined by means of gestUI is the input to this procedure.
- The gesture catalogue is imported in the framework selected to test the gestures.
- The user employs the gestures defined to do some tasks in the framework.
- If each gesture satisfies the requirements of the user, then the gesture is confirmed in the catalogue.

- Else, the user requires the definition of a new gesture in order to replace the gesture that doesn't satisfy the requirements of the user.

4 Capability Design Tool (CDT)

The tool support for design and runtime of solutions based on Capability Driven Development (CDD) is called Capability Design Tool (CDT) (see Fig. 4, left), it is designed as an integrated development environment built on the Eclipse Framework, using the Eclipse Modelling Framework (EMF) technologies. This tool supports capability modelling according to the capability metamodel including context modelling and goal, process and concepts models.

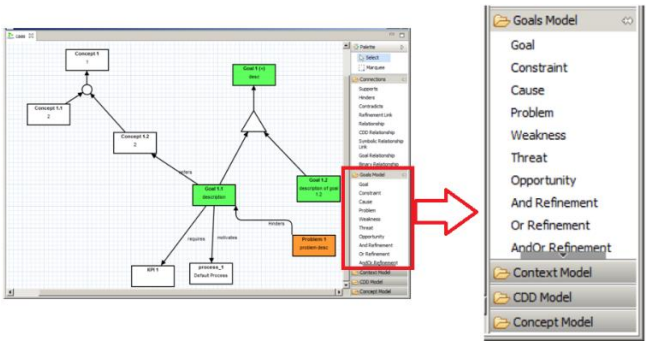

Fig. 4. CDT (left) and Goal Model palette (right)

CDT permits draw goal model, context model, CDD model, concept model. In this paper, we consider goal model in order to define the gesture catalogue to apply gestUI to define gestures. The elements of the palette corresponding to the goal model are shown in Fig. 4 (right).

5 Applying the method

gestUI can be applied in any information system with the aim of include gesture-based interaction in the user interface. In this paper, we consider CDT in order to apply this method to include gesture-based interaction in its user interface to draw goal diagrams.

The first step is the definition of gesture catalogue. In Table 1 we show an excerpt of the gesture catalogue definition in order to apply gestUI to draw goals models in CDT.

Table 1. Excerpt of gesture catalogue

Element	Action	Gesture	Element	Action	Gesture
Goal	create Goal		U	create Cause	
Constraint	create Constraint		P	create Problem	

Therefore, employing the procedure specified in the first method component the user defines a set of gestures in order to execute actions in CDT. Then, using a user interface, the user employs his/her finger to sketch strokes which define a gesture, see Fig. 5.

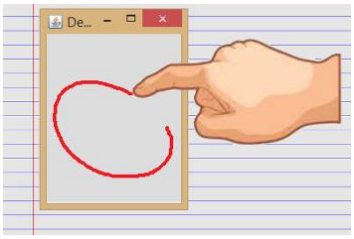

Fig. 5. Gesture sketched by the user

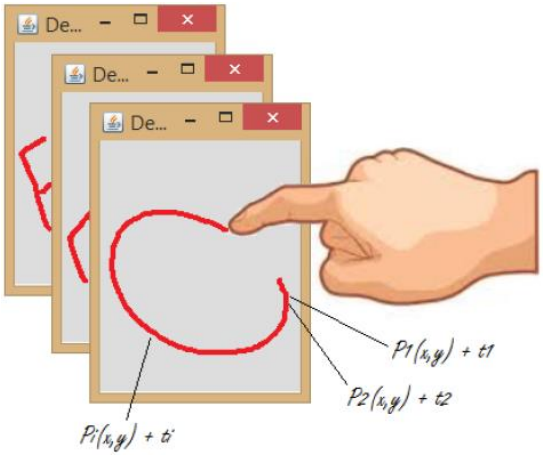

Fig. 6. A gesture catalogue

Then, according to the description of this method component the output of the procedure is the set of gestures conforms to the gesture catalogue model (see Fig. 6),

each gesture is defined as a set of strokes with points defined by coordinates (X, Y) and the corresponding timestamp.

Then, considering as input this gesture catalogue model and applying a model transformation we obtain a platform-specific gesture catalogue. In Fig. 7 is shown an excerpt of set of gestures according to the initial definition of gestures.

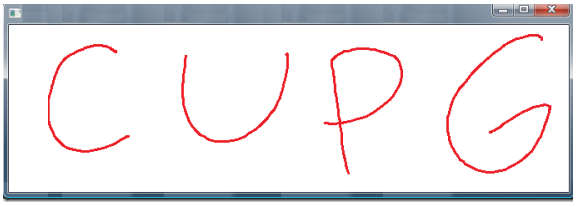

Fig. 7. A platform-specific gesture catalogue

The next method component is used to define the gesture-based interaction by means of the specification of the correspondence gesture-action, which is dependent on the function of the information system. In Table 1, it is specified the actions to execute using gestures defined.

The last method component permits the automatic generation of the source code including the gesture-based interaction with the customised gestures. In this case, we require as input the source code of the user interface, the specific-platform gesture catalogue. Then, the user select a gesture contained in the catalogue and an action defined in the user interface source code in order to define the correspondence gesture-action.

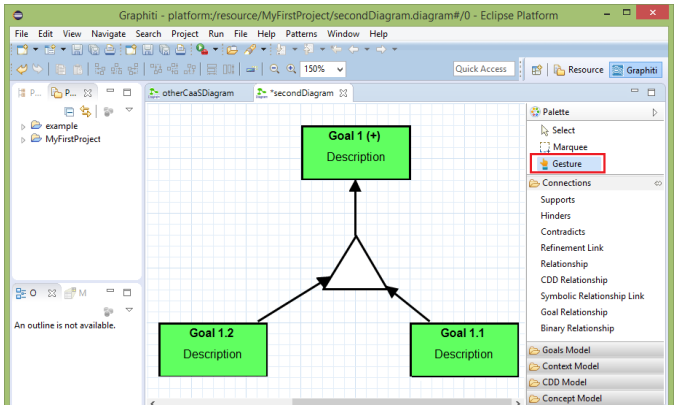

Fig. 8. Applying the gesture-based interaction in CDT

Finally, the process has finished and the CASE tool called Capability Design Tool (CDT) has included the gesture-based interaction which can be used to draw goal models (Fig. 8).

6 Conclusions and Future Work

gestUI, a model-driven method for developing multi-stroke gesture-based user interfaces is described in this paper using the "Template for the Documentation of Method Components in CaaS". We demonstrate the application of the method and supporting tools in the CASE tool called Capability Design Tool (CDT) which is built using Graphiti (an Eclipse framework with the structure of a plug-in) that is used to draw diagrams. We produced the 'Platform-specific gesture specification' for this CASE tool in order to illustrate the multiplatform capability of the approach. The gestures were successfully recognised using $N as a gesture recognizer. Then, we automatically generated the final gesture-based interface components and integrated them into the user interface.

Some advantages of gestUI are: (i) its platform independence enabled by the model-driven development paradigm, (ii) the convenience of including user-defined symbols and (iii) its iterative and user driven approach.

Future work will be developed along the following lines: (i) to include a feature to that the user can define gestures according to his/her preferences during the execution of the information system, (ii) developing a Technical Action Research in order to validate this method in the "Capability as a Service for Digital Enterprises" Project (CaaS project).

Acknowledgments

Otto Parra is grateful to his supervisors Sergio España and Óscar Pastor for their invaluable support and advice. This work has been supported by Secretaría Nacional de Educación, Ciencia y Tecnología (SENESCYT) and Universidad de Cuenca of Ecuador, and received financial support from Generalitat Valenciana under Project IDEO (PROMETEOII/2014/039).

References

[1] F. Karray, M. Alemzadeh, J. Abou Saleh and M. Nours Arab, "Human-Computer Interaction: Overview on State of the Art," *International Journal on Smart Sensing and Intelligent Systems,* vol. 1, no. 1, pp. 137-159, 2008.

[2] M. Hesenius, T. Griebe, S. Gries and V. Gruhn, "Automating UI Tests for Mobile Applications with Formal Gesture Descriptions," *Proc. of 16th Conf. on Human-computer interaction with mobile devices & services,* pp. 213-222, 2014.

[3] S. H. Khandkar, S. M. Sohan, J. Sillito and F. Maurer, "Tool support for testing complex multi-touch gestures," in *ACM International Conference on Interactive Tabletops and Surfaces, ITS'10*, NY, USA, 2010.

[4] O. Parra, S. España and O. Pastor, “A Model-driven Method and a Tool for Developing Gesture-based Information Systems Interface,” in *Proceedings of the CAiSE'15 Forum at the 27th International Conference on Advanced Information Systems Engineering*, Stockholm, Sweden, 2015.

[5] J. Grabis, L. Jokste, G. Bravos, J. Stirna, T. Gonzales, M. Henkel and H. Koc, “Capability Modeling: Initial Experiences,” *Perspectives in Business Informatics Research: 13th International Conference (BIR 2014)*, pp. 1-14, 2014.

[6] J. Stirna, J. Grabis, M. Henkel and J. Zdravkovic, “Capability driven development - An approach to support evolving organizations,” in *5th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2012)*, Rostock, Germany, 2012.

[7] K. Sandkuhl and J. Stirna, “CaaS Base Methodology,” UR, 2014.

[8] K. Sandkuhl and J. Stirna, “Template for the Documentation of Method Components in CaaS,” 2013.

[9] S. España, T. González, J. Grabis, L. Jokste, R. Juanes and F. Valverde, “Capability-driven development of a SOA platform: a case study,” *First International Workshop on Advances in Services DEsign based on the Notion of CApability (ASDENCA 2014),* vol. Springer LNBIP 178, pp. 100-111, 2014.

[10] J. Vanderdonckt, “A MDA-Compliant Environment for Developing User Interfaces of Information Systems,” *Advanced Information Systems Engineering LNCS in Computer Science,* vol. 3520, pp. 16-31, 2005.

[11] J. Vanderdonckt, “Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges,” in *ROCHI'08*, Iasi, Romania, 2008.

[12] J. Coutaz and G. Calvary, “HCI and Software Engineering for User Interface Plasticity,” in *The Human-Computer Handbook. Fundamentals, Evolving Technologies, and Emerging Applications*, Julie, A.; Jacko, ed. CRC Press Taylor and Francis Group, 2012, pp. 1195-1220.

[13] G. Calvary, A. Dery-Pinna, A. Occello, P. Renevier-Gonin and M. Riveill, “At the Cross-Roads between Human-Computer Interaction and Model-Driven Engineering,” *ARPN Journal of Systems and Software,* vol. 4, no. 3, pp. 64-76, 2014.

[14] A. de Carvalho Correia, L. Cunha de Miranda and H. Hornung, “Gesture-based interaction in domotic environments: State of the art and HCI framework inspired by the diversity,” in *INTERACT'13*, Barcelona, Spain, 2013.

[15] R. Picek, “Suitability of Modern Software Development Methodologies for Model Driven Development,” *JIOS,* vol. 33, no. 2, pp. 285-295, 2009.

[16] A. Kleppe, J. Warmer and W. Bast, MDA Explained: The Model Driven Architecture : Practice and Promise, USA: Addison-Wesley Prof., 2003.