A Framework for Modelling the User Interaction with a Complex System

Mª L. Rodríguez Almendros, Mª J. Rodríguez Fórtiz, and M. Gea Megías
GEDES Group. Granada University (SPAIN)
{mlra,mjfortiz,mgea}@ugr.es

Abstract. Nowadays, the increasing performance of modern computers enables interactive systems focus on the graphical handling of information. This increase in computational power allows the designer to develop better interfaces oriented to more intuitive human-computer interaction using an interaction style more adapted to the user and system characteristic. However, when the interaction is more intuitive for the user, the design and implementation phase requires great effort by the designer. In this paper, we propose a suitable framework to analyse and design these complex systems, and a formal model that allows us to prove the system properties and validates the specification.

1 Introduction

The increasing performance of modern computers enables interactive systems focus on the graphical handling of complex systems. This increase in computational power allows the designer to develop better interfaces oriented to more intuitive human-computer interaction. Interaction style should be adapted to the user (skills, age, education, necessities, etc.) and system characteristics (components, functionality, etc.).

Interactive systems based on the desktop metaphor have adopted direct manipulation as an interaction style. This style proposes a visual model for the underlying information based on objects and actions by using input devices [1]. An interactive system based on a direct manipulation style is characterised by a user-friendly interface offering a natural representation of objects and actions [2]. This style is characterised as follows:

• Continuous representation of objects. Application objects have a graphic representation, which is visible to the users. This graphic appearance represents the object interface, and it provides information about its state (selectable, visible, active, etc.).

• Physical actions over objects. The user directly manages these objects by using input devices such as the mouse or keyboard. In this style, pointing and clicking replace the writing. Each object has certain handling capabilities (movement, dragging, dropping, etc.) that can be performed by using these input devices.

• The semantics of the action is context sensitive. An action could be carried out on several objects. Depending on the kind of object, the type of action and the gesture, the meaning of the action could be different. For example, the meaning of the
• action "dragging and dropping" could be "copy", "erase" or "print", depending on the selected object and the location where it has been dropped (a folder, the trash bin, a printer).

Direct manipulation depends on visual representation of the objects and actions of interest, physical actions or pointing instead of complex syntax, and rapid incremental reversible operations whose effect on the object of interest is immediately visible. This strategy can lead to interactive systems that are comprehensible, predictable and controllable [3]. These features represent design constraints (e.g., in task analysis) imposed by the interaction style. A formalisation of such concepts may help the designer to understand the possibilities of such an interaction style in order to achieve better applications. The direct manipulation style has traditionally been identified with the desktop metaphor environment with drag-and-drop facilities. But other approaches such as Virtual Reality, Augmented Reality or Ubiquity Computing are based on the Direct Manipulation Style, changing the space and rules by which the object of interest is shown (input and output of the traditional computer) and managed (grasping and dropping virtual objects).

These complex systems require complex graphical applications which implement:
• The interaction using new input devices and sensors to collect the user information.
• Modelling of several kinds of objects with different behaviour, features and properties.
• Management of complex interactions between objects based on spatial relationships, comparison of features, hierarchies (part-of or is-a relationships) and the way of user interaction.
• High level formal specification techniques and languages to facilitate the design of user manipulation. These techniques should also be used to analyse and verify the properties of the system in each moment.

We propose a framework [4], [5], [6] of interactive systems based on a direct manipulation style focusing on relevant aspects of this interaction model from the user’s point of view. This approach allows us to describe formally the system objects, their relationships and the interaction (the user’s manipulation process) with them. Our model allows us to analyse usability principles of the interactive systems based on a direct manipulation style. It also defines and checks properties of usability that every system must possess, which are deduced from the formal representation of the model. We have also design a specification method, a language, and a prototype based on this model that allow the developer to define and test the elements of the manipulation system, the relationships and the properties.

Next section will present a formal model of interactive system based on a direct manipulation style, focusing on objects and their relationships, graphical representation, interaction facilities and formal properties of this kind of interactive systems. We will propose a specification method and will present a formal language for specify the manipulation process of an user on a complex system in section 3. We will finish with conclusions section.
2 A Formal Model Based on a Direct Manipulation Style for Complex Systems

The formalisation must take into account the following elements:

- **Objects**: Taxonomy which represents the system components with handling capabilities. The system is a set of objects.
- **Functionality**: The objects carry out actions dynamically according to the user interaction.
- **Relationships**: Sets of associations between objects due to the user interaction, represent the degree of freedom for objects handling.
- **Manipulative processes**: Describe the system actions through manipulation of the objects.

2.1 Objects and System

The objects can be characterised by the following aspects:

- **Features**: The features describe the relevant characteristics of an object. They depend on the object domain. Objects have spatial features (position, orientation, etc.), visual ones (colour, appearance, etc.), modes (selected, dragged, etc.), etc. Each object has a set of features whose values determine the state of the object at a given time. The change in features may represent a change in state.
- **Physical representation**: Each object has a visual representation, by which it defines its external behaviour. This graphical representation constitutes the object interface and it gives information to the user about the underlying object and its internal state.
- **Actions**: Each object has functionality. Some actions are directly attached to the user, who manipulates the object by using input devices (mouse, keyboard, datagloves, etc.). These user actions change the object features (size, position, etc.). Actions can be also triggered by the object itself or by another object.

The application is composed of objects. The concepts of object and object domain are defined as follows:

**Definition 1**: An object is an element of an application with identity, features and processing capability. The object domain \( O \) is the set of all these elements.

Thus, we also consider the input devices as part of the object domain denoting the available object set for user handling. Therefore, these devices are included in the object domain as a subset of the latter called the control object set \( A \).

The internal state of an object is given by its relevant features. Therefore, we represent the set of features for the \( i \)-object, \( P_i \). A function is defined for each one the features:

**Definition 2**: The feature function \( p_{i,k} \), describes the current values of k-property for the \( i \)-object.

The values can be integers, characters, booleans, enumerate sets, etc. Each feature of set \( P_i \) has a value, and these values determine the object state. This relationship is defined by the following function:
Definition 3: The state-feature function, \( f_i,p \), for an object describes the state \( p \) for the object \( i \) based on its features.

Every object has a state at any given moment. We have to know the object state at each instant of time, and therefore, the following function is defined:

Definition 4: The state function at the instant \( t \), \( e_i \), describes the object state of each object \( i \) at the instant of time \( t \).

The system (as a whole) has a state at any given moment (describing its behaviour). This state is composed of the states of each one of its objects. Therefore, it is defined as follows:

Definition 5: The system states are constituted of all the possible states of the system.

Definition 6: An image, \( /G_{ij} \), is a visual representation of the \( i \)-object in the \( j \)-state.

As commented above, every object has an associated visual representation. This representation allows us to handle the object, and it gives us information about its internal state. The image domain \( \Omega \) defines the set of elements belonging to the system interface. To avoid ambiguity, for each element in the \( O \) set, there should be only one related visual representation or image.

The relationships between objects and their visual representation can be described with the following mapping:

Definition 7: The representation function, \( \rho \), is a mapping function from objects to images.

This kind of relationship between an object and its visual representation is also known as feedback, that is, the observable effect of an object through the display of an interactive system. But in this context, the user not only "sees" the objects, but in fact, can interact with them. Each interface object can be freely handled, and these changes in the interface (domain) produce changes in the underlying system. Each object has a related functionality within the system. In other words, the objects provide a set of available actions that may be performed on them. When an action is performed on a set of objects, this implies a state change on the objects.

The objects of the system have handling capabilities. The user interacts with the system by carrying out actions on the objects. When the user performs an action on a set of objects, the action implies a state change on the objects, and therefore a state change on the system. This concept is formalised in the following definition:

Definition 8: An action, \( \alpha \), on the system is a function which changes the system state.

We denote as \( A \) the set of actions over the system. The set of actions on the system determines the system's functionality, that is, the actions that the user can carry out on the system.

The direct manipulation style represents an alternative method to access the system functionality. The user directly handles the interface objects (icons or images) to perform the user task. Therefore, the handling performed on the interface objects implies a change in their features (such as change of position, colour, size, etc), and also in their state. Therefore, we identify these processes as a special kind of actions on the system, and we describe them as gestures.

Gestures are performed by using input devices (control objects). In this process, several gestures (or devices) may be applied to obtain the same result. Therefore,
more than one gesture may be attached to a common action. For example, several gestures are provided for closing a window (pointing and clicking on the close option, typing a control sequence, etc.). Thus, for any given system action, more than one gesture may exist, and this property is expressed as follows:

**Definition 9:** A *gesture*, $g_{n,s}$, over the system for a system action is a function defined in the image domain expressing a handling action on the interface.

The above notation for a gesture, $g_{n,s}$, means that a gesture identified as $g$, is related to the system action $n$, and more than one gesture may exit for such an action. The gesture domain is denoted by $G$, the set of gestures over the system. Note that some of these gestures may be as complex as we wish. For example, dragging an image is a selection task, which involves a mouse movement, focussing on the image, pressing the mouse button, moving it and releasing the button. These interaction activities represent a change of the image location, and thus, may be denoted as a single gesture, $G_{drag}$.

The occurrence of a gesture implies a change in the objects themselves. The user has changed its visual representation, and therefore, the objects have been involved in a transformation action. By changing the interface objects we change the domain objects. The interface objects constitute the way of performing actions on the system, and therefore this feature characterises the direct manipulation paradigm. Therefore any gesture related to a system action. We may explain this relationship by the following mapping function, which relates gestures to system actions:

**Definition 10:** The *manipulation function*, $\mu$, defines a mapping from the gesture domain in the actions domain.

A direct manipulation style allows the user to perform an action in different ways. Therefore, more than one gesture may be related to a certain action. The inverse is not true, however; a gesture can only perform one action at any given time.

### 2.2 Direct Manipulation System

As a consequence of the previous definitions we may characterise such system as follows:

**Definition 11:** A *direct-manipulation system* $S_{dm}$ is defined as:

$$
S_{dm} = \langle O, \Omega, e, \rho, A, \zeta, \mu \rangle
$$

where $O$ represents the object domain, $\Omega$ is the image domain, $e$ is a function representing the system state in the instant of time $t$, $\rho$ is the relationship between objects and their representations, $A$ and $\zeta$ are functions representing the system actions and gestures respectively, and $\mu$ is a function mapping gestures with respect to actions.

### 2.3 Properties of a Direct Manipulation System

Our model allows us to represent the formal properties of this kind of interactive system. For example, it may be used to analyse usability principles [7], [8], [9], [10].
The set of properties makes it easy to achieve specified goals effectively and efficiently while promoting user satisfaction. Some such goals are:

- Observability. Visibility of the system state. The user should know the system state from the feedback.
- Matching of system and real world (use the same facts and concepts...)
- Flexibility. The user may have more than one way of doing something
- Consistency. The user should be able to predict the system response in similar situations.
- Predictability. The user's knowledge of the interactive history is sufficient to determine the result of the next interactions.
- Coherence. Any change performed in the interface is immediately applied to the corresponding objects.
- Transparent. Any system behaviour is observable from the user interface.
- Fully accessible. Any system action is accessible by the user interface using gestures.

Some of these properties can be verified within the direct manipulation system.

**Definition 12:** A direct-manipulation system is *consistent* if the objects features identify only one state at any time.

The consistency property is closely related to unambiguity. Therefore, it is important to guarantee that the system state is directly obtained from the object states (by the feature values) at any time.

**Definition 13:** A direct-manipulation system is *predictable* if each system state associates only one image for each object.

It means that the user's knowledge of the interactive history is sufficient to determine the result of the next interactions. The user determines the system state by observing the image of its objects, which represent the object state at each moment. In this way, when the user interacts with the objects, he or she can predict the following state without ambiguity. Note that the correct choice of the graphical metaphor is essential for recognition and recall.

**Definition 14:** A direct-manipulation system is *coherent* if any change performed in the interface is immediately applied to the corresponding objects.

An interactive system should always keep the user informed about what is going on, through appropriate feedback within a reasonable time. This property ensures that the interface handling effectively allows the user to have access to the system functionality. Therefore, a gesture implies a change in the set of objects deduced from a change in their images.

**Definition 15:** A direct manipulation system is *transparent* if any system behaviour is observable from the user interface.

Observability allows the user to evaluate the internal state of the system by means of its perceivable representation from the interface. This property is important because it means that the system is not viewed as a black box, and therefore, the user interface allows the user to change any feature in the object domain.
Definition 16: A direct-manipulation system is *fully accessible* if at least one gesture associated with each system action actually exists. It is related to the multiplicity of ways the user and the system may exchange information.

These properties guarantee some desirable properties of an interactive system based on a direct manipulation style. For example, it is easier to learn (learnability) and use (efficiency) if gestures are well defined. Moreover, these conditions could be imposed in the design phase as a necessary condition to achieve system usability. This theoretical analysis could help the designer in the creation of the user interface and its metaphors, guiding design decisions and prototype development. These principles can be expressed as follows:

Definition 17: A direct-manipulation system has the *property of usability* if the above properties are satisfied, that is, it is consistent, predictable, coherent, transparent and fully accessible.

Note that these properties improve system usability, but this is not a sufficient condition. Formal methods aid in the design phase, but may not ensure system usability because they are directly related to the user's evaluation. However, the formalisation may help to enhance relevant properties directly related to the user's point of view.

2.4 Study of Relationships

In a direct manipulation system the user has access to the system functionality (system actions) handling the objects. These handling processes (gestures) involve relationships among the objects of the system.

The gestures represent the syntax for direct manipulation actions and the relationships represent the semantic for these actions. Gestures produce new relationships among the objects, for example a new localisation of the objects into the space (be inside of, out of, etc.), or new values of features (an object changes the colour when it is selected, etc.). We define the following types of relationships: topological, gestures and property.

Definition 18: A *topological relationship*, $\mathcal{R}_{\text{topological}}$, is a relation between images given by their spatial location.

The set of possible relationships that could appear is close related to the possible spatial distribution of objects.

Gestures can be treated as relationships between images and control objects.

Definition 19: A *gesture relationship*, $\mathcal{R}_{\text{gestures}}$, is a binary relationship between the control object domain ($\Delta$) and the images that gives information about whether one gesture has been done with a particular device.

Analogously, properties between objects can be treated as relationships. Any common property shared by several objects can be inquired as a relationship.

Definition 20: A *property relationship*, $\mathcal{R}_{\text{property}}$, is a relationship between objects, which gives us information about whether one common property holds.

This notation allows us to express any system event as a set of relationships between objects using a high level of abstraction. This characterisation is powerful enough to treat low-level events as well as topological relationships as object
properties caused by their handling: the former by direct manipulation and the latter as consequences of the handling.

Topological relationships, gesture relationships and property relationships are defined in the object domain and the set system relationships, as follows:

**Definition 21:** The set of system relationships, $\mathcal{R}_s$, is composed of gesture relationships, topological relationships and property relationships defined in the system for the objects.

### 2.5 Syntactic Direct Manipulation Model

**Definition 22:** A syntactic direct manipulation model, $M_{mm}$, is composed by the tuple

$$M_{mm} = (S_{mm}, \mathcal{R}_m)$$

where $S_{mm}$ is a direct manipulation system and $\mathcal{R}_m$ is the set of system relationships (topological, gesture and property) defined in the direct manipulation system.

The user interacts with the objects in the system through gestures, new relationships between the objects are produced, changing the state of the system due to the modifications produced in the states of objects. The direct manipulation system and the relationships between the objects change with the time.

### 3 Specification Methods and Language

Once a formal model based on a direct manipulation style has been provided, a specification method is supplied to obtain relevant properties of such a system and to describe the manipulation process. When a gesture is produced in the system, an action is carried out. Specifying a manipulation process means describing the objects, the relationships and when and how the system also have to be taken into account. The object relationships and their logic and temporal dependencies have to be analysed. This allows us to identify the action to be carried out in the system, for which we define a process called the manipulative process.

**Definition 23:** A manipulative process, $p_m$, is a set of objects and relationships (relation-formulas) which describes an interaction with the system.

The relation-formulas are used to specify when the user can carry out an action. The set of manipulative processes in the system is denoted as $P_m$.

**Definition 24:** A relation-formula, $f_r$, is a finite sequence of topological, gesture and feature relationships connected by operators, which represents a condition over the system state.

We have defined tree types of operators: logical, temporal and recursive. To interpret a relation-formula, the state of the system is considered. Logical and temporal interpretations of operators and formulas are based on [11].

Therefore, based on the syntactic direct manipulation model $S_{mm}$ and the set of manipulative processes $P_m$, we propose a specification method, MOORE, oriented to objects, relationships and manipulative processes.
Definition 25: The specification model of a direct manipulation system, $S_{OMR}$, is defined as the tuple

$$S_{OMR} = (M_{MMDS}, P_M)$$

where $M_{MMDS}$ is a syntactic direct manipulation model and $P_M$ is the set of manipulative processes in the system.

The domain of the manipulative processes $P_M$ represents the semantics of the system whereas the syntactic direct manipulation model $M_{MMDS}$ is the syntactic part.

First, we define the objects and the relationships, and last we describe the interaction process between the user and the system, the manipulative processes.

We have also defined a specification language, MORELES, based on objects, relationships and manipulative processes. A specification of an interactive system using this language is divided in several modules:

- One module to specify each objects domain.
- One module to specify the system.

The module which specifies the system imports the objects modules and includes information about the objects interactions from different domains and conditions to carry out them. This division in modules facilitates the specification of the complex system, provides independence and allows the reutilisation of the specification of an object domain in different systems.

Besides, each module is divided into sections as we show:

- Object domain module: This module describes the control objects, features, states, graphical representation, actions, gestures, topological and manipulative processes of the objects of this domain.

- System module: The module that specifies the system defines objects domains, actions and gestures of the system, topological and properties relationships and manipulative processes.

Moreover, we have generated a Java prototype automatically from the specification language. A class is created for each object domain. Another class implements the system functionality and structure. Java facilitates the management of graphical representations (images) and the modeling of gestures, treating them as events.
4 An Example: Set Game

In this section we provide an example from a simple case study to illustrate how the formalism, the specification method and language we have introduced are used, showing their expressiveness. Many applications are based on spatial relationships among objects. As examples are the kid games. We have selected these examples because the direct manipulation of objects by the child is crucial for his learning process, and thus, the knowledge is given by rules (ordering, associations, etc.).

The set game is an easy game, which allows us analyse the child capacity to make associations (belong relationships). The rule for the game is grouping together the game pieces that have a belong property. For example, the set game has tree objects domain: mouse, dogs and doghouses. The objective of this game is to include a dog in doghouse. The specification of the game is divided in four modules, one module to specify each object domain (mouse, dogs and doghouses) and one module to specify the system (set game).

To illustrate the use of the language, only the objects domain dogs is defined:

```plaintext
domain dogs;
//Control objects
import mouse;
//Attributes of the objects
features position (x,y), mode (normal, selected);
//States of the objects
states state_normal = ( position (x,y), mode (normal)),
  state_selected = ( position (x,y), mode (selected));
//Graphical representations and states
graphical_representation state_normal → image_normal,
  state_selected → image_selected;
//Actions of the objects
actions
  select: state_normal → state_selected,
  move(position,mode):(state_selected((x,y),mode:selected) →
  state_normal((c,d), mode:normal);
//Gestures relationships of the objects
gesture_relationships ℜselect(dogs), ℜmove(dogs);
//Gestures relationships and actions
actions_gestures select=(ℜselect), move=(ℜmove);
//Topological relationships
topological_relationships (mouses on dogs);
//Manipulatives processes
manipulative_processes
//Manipulative process: Select a dog
ℜselect(dogs):= ∃! d ∈ domain(dogs), ∃! r ∈ domain(mouses):
  (r on d) ∧ ℜclick(r),
//Manipulative process: Move a dog
ℜmove(dogs) := ∃! d ∈ domain(dogs), ∃! r ∈ domain(mouses):
  ℜdrag(r) ∨ (r on d);
```

The control object *mouse* is used to manipulate the objects *dogs*. The dogs have two feature (*position* and *mode*), two states (*normal* and *selected*) and therefore two graphical representations. The user can *select a dog* and *move a dog* (actions and gestures *select* and *move*). This domain has a topological relationships defined, *on*, with the control objects domain, *mouses*.

Two manipulative processes have been defined: *Select a dog* implies that the mouse is over the dog (topological relationships, *on*) and the mouse button is pressed (gesture relationships, *click*); *Move a dog* implies that the mouse is over the dog (topological relationships, *on*) and the mouse is dragged to a new position (gestures relationships, *drag*), and while the mouse is being dragged the mouse is over the dog (operator \( S \land \), which means *since and*).

5 Conclusions and Future Works

A framework of specification of complex interactive systems has been presented. It allows to model the components of a direct manipulation system, the objects, and the relationships between them. In order to allow the manipulation of the objects, actions, gestures and conditions (relation-formulas) may be specified.

The framework may be used to analyse the usability properties of the interactive system: consistent, predictable, coherent and transparent. As the system is specified formally (set theory and logic are used to describe its architecture) its properties can also be verified formally, using the same formalisms.

To facilitate to the user the description of the system, its properties and the manipulation process, a specification language has been proposed. The syntax of a specification written with this language can be translated automatically to a Java prototype. It allows that the user verifies and validates the system.

Our future work is the extension of the framework to specify the manipulation of several users at the same time. It is useful to model the interaction of a group of users in the cooperative and collaborative systems. Cooperative models and concurrency are been taken into account.

References