Towards Spatial Specification of Interactive System

M.L. Rodriguez, M. Gea, F.L. Gutiérrez
Dpt. Lenguajes y Sistemas Informáticos, ETSI Informatica, Universidad de Granada, 18071 Granada, Spain.

email: <mlra, mgea, fgutierr>@goliat.ugr.es

Abstract

Modern interactive systems are oriented towards information management in a graphical environment. Formal specification techniques [1] *help designers to describe systems, focusing on the relevant aspects of the interaction model. We analyse here a new way of describing the specifications of a direct-manipulation system and we propose an extension of this concept to cover the specifications of a system based on spatial relationships.*

Keywords: formal methods, direct-manipulation style, spatial relationships.

1. Direct-Manipulation Style

Nowadays applications are being developed for operating systems based on a graphical environment and tend to be based on the direct-manipulation paradigm [2], which is characterised as follows:

- ***Continuous representation of objects***. Application objects have a graphic representation, which is visible to the user. This graphic appearance represents the object interface and gives information about its state (selectable, visible, active, etc.).
- ***Physical actions over objects***. The user directly controls these objects via the mouse or keyboard, which have certain manipulative capabilities (movement, dragging, dropping, etc.) in relation to actions performed with control objects.
- ***The action semantic is context sensitive.*** In fact, any action that can be performed with objects is implicit in the interactive activity itself. The meaning of to drag and drop objects depends upon the objects selected and where they are to be dropped (copy a file if it left in a folder, erase if it is left in trash, etc.).

Modern operative systems based on the desktop metaphor have successfully adopted the direct manipulation concept as an interactive style. These concepts have been extended to programming languages to create new applications based upon this paradigm. But one drawback is the limitation of this concept. These applications are event oriented, and the available events are restricted to object movement (drag and drop) and inclusion testing (inside another component). A problem arises however, when we attempt to design more complex behaviour based on spatial relationships (to the left of, behind, between...).

We consider that the direct-manipulation style remains valid, but it is necessary to extend the concept of spatial interactive environments to embrace more complex relationships between objects. This is evident because the traditional indirect-pointing device (the mouse) is slowly evolving into new devices (datagloves) and techniques (gesture recognition [3] and VR [4]). Within this context information management is becoming clearer and more intuitive for the user, but it increases the complexity of implementation for the designer. The specification technique helps system description by focusing on the relevant aspects of model interaction.

This paper, therefore, focuses on a spatial specification model and in the next section we specify these spatial relationships, and conclude with examples of such systems.

2. A Direct Spatial-Manipulation Style

We propose to extend the features of a direct-manipulation style to embrace complex (spatial) relationships between objects. To do this the following concepts must be taken into account:

- ***Objects*** represent the carrier set with manipulative capabilities. Objects represent application concepts.
- ***Relationships*** represent the degree of freedom for object manipulation. Relationships can be classified as *gestures* or *topologies*. Gestures define the object manipulation capability (press, release, move, etc.) with an intuitive

Published by the Eurographics Association ISSN 1017-4656

meaning, while topological relationships denote spatial relationships among objects (within, outside, etc.). Gestures define object behaviour while topological relationships define system behaviour.

- ***Dynamics of the actions***. The (system) actions are triggered by the occurrence of topological relationships caused by a gesture.

Other specification techniques, such as State Transition Diagram [5] or UAN [6] have been used. STN is focused on a state-based approach, and objects are not explicitly represented. The UAN notation uses a similar concept but is focused on describing a task and only considers a limited range of topological relationships. In the following sections these components will be discussed.

2.1. Objects

Objects are components with manipulative capabilities representing application concepts (entities, attributes, abstractions, etc). Objects and their visual representation are usually coupled, but we shall consider a distinction in order to study the properties of a direct-manipulation style.

An *object* is an element of an application with identity and processing capability. The object domain ***O*** is the set of all these elements. Analogously, an *interface object* is a compound of an interactive system with a graphical representation. The interface object domain Ω defines the set of elements belonging to the system interface.

The representation function, ρ is a mapping function from objects to interactive objects, with the following properties:

$$\rho : O \rightarrow \Omega$$
$$\forall \upsilon \in \Omega, \exists_1 o \in O / \rho^{-1}(\upsilon) = o$$

The constraint imposed upon the graphic representation is that each interface object identifies only one application object. Otherwise, ambiguity arises in the manipulation of the interface.

System behaviour is defined by the dynamic of the objects. The set of object *actions* is represented as a function, ***A***, over the object domain and defined as follows:

$$A: O \rightarrow O$$

The direct-manipulation style is an alternative method to access the object functionality. The user directly manages the interface objects to perform the user task. The manipulation performed on the interface object by direct manipulation implies a change to the related object, such as its position, status (open, closed, selected), properties (deleted, changed), etc. Therefore, we identify the interface actions as special kinds of controls in the system, and we describe them as gestures.

A *gesture*, ς, is a function applied to interactive objects representing a manipulative action, and we denote as ζ the set of gestures domain.

$$\zeta : \Omega \rightarrow \Omega$$

In direct manipulation a close relationship may exist between (object) actions and gestures.

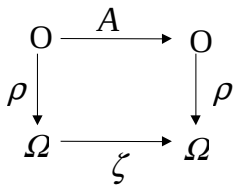

Figure 1 . The direct-manipulation diagram

In fact, a ***direct-manipulative system*** *S* is defined as follows:

$$S = <O, \Omega, \rho, A, \zeta, \mu>$$

where O represents the object domain, Ω is the interface object domain, ρ is the relationship among objects and their representations, A and ζ are functions representing the system actions and system gestures respectively and μ is a function mapping actions to the respective gesture

$$\mu: A \rightarrow \zeta$$

Ideally, a direct-manipulation system should allow the user to carry out any action in the system by gestures, as long as the μ function is injective.

2.2. Relationships

If we take it that a gesture represents the syntactic level of an action, a *relationship* represents the semantic level of a direct manipulation gesture. That is, a g*esture* may produce a new localisation of the objects in the space, whereas its meaning is given by a set of relationships among the objects. These relationships can be as complex as we want. One of the most relevant kind of relationships is given by the spatial localisation of object (represented by a gesture), called topological relationships.

A ***topological relationship*** $\Re_t$ is a relationship among objects obtained from their spatial location and can be expressed as follows.

$$\Re_t : \Omega \ , \Omega \ \rightarrow \text{Boolean}$$

For example, inclusion is a topological relationship between two objects described as follows:

$$\Re_{in} : \Omega, \Omega \rightarrow \text{Boolean}$$
$$\Re_{in}(\upsilon_i, \upsilon_j) = \text{True} \Leftrightarrow \upsilon_i \subseteq \upsilon_j, \quad \upsilon_i, \upsilon_j \in \Omega$$

The set of possible relationships which may result is intimately related with the possible spatial distribution of objects. A notation to describe possible spatial relationship between objects is needed. The topological domain may be as complex as required. For example, the following relationship set is defined in a two dimensional space.

$$\Re^{2d} = \{ \leftarrow, \uparrow, \rightarrow, \downarrow, \sim \}, \ \Re^{2d} \subset \Re_t$$

$o_i \leftarrow o_j = \{ \forall p(x,y) \in o_i, \forall q(x,y) \in o_j : p.x < q.x \}$
$o_i \rightarrow o_j = \{ \forall p(x,y) \in o_i, \forall q(x,y) \in o_j : p.x > q.x \}$
$o_i \uparrow o_j = \{ \forall p(x,y) \in o_i, \forall q(x,y) \in o_j : p.y > q.y \}$
$o_i \downarrow o_j = \{ \forall p(x,y) \in o_i, \forall q(x,y) \in o_j : p.y < q.y \}$
$o_i \sim o_j = \{ \forall p(x,y) \in o_i, p \in o_j \}$

A gesture may be treated as a relationship between an object and a control object (a mouse for example). Therefore, a gesture relationship can be defined as follows:

$$\Re_g : \Delta, \Omega \rightarrow \text{Boolean}$$

where Δ is a control object, and Ω is the interface object. It can also be defined as monadic $\Re_g(\Omega)$ if the control object is not relevant. Extending this notation, object properties can also be treated as a relationship, inquiring whether these objects satisfy this property.

$$\Re_p : O, O \rightarrow \text{Boolean}$$

Relationships imply actions. When a relationship occurs, an action must be performed. The application semantic is attached to actions triggered by topological relationships (called rules). The **specification** of a spatial manipulative system $S = <O, \Omega, \rho, A, \zeta, \mu>$ consists of a set of rules as follows:

$$\Re \Rightarrow A$$

The relationship $\Re$ may be as complex as we want, mixing gestures, topological relationships and object properties. In the next section we describe some examples of specifications using this formalism.

2.3. Evolution

An important aspect is the dynamic evolution of the interactive system and thus the evolution of the relationships that are contained therein. Gestures usually result in new relationships being satisfied as a consequence of the gesture operation.

A relationship $\Re_a$ induces the relationship $\Re_b$, noted as $\Re_a \Rightarrow \Re_b$, if the truth of $\Re_a$ is necessary to satisfy $\Re_b$.

For example, an inclusion relationship is usually incorporated by a dragging gesture by the user. The user picks the object, drags it and drops it onto another object. This is characterised as follows:

$$\Re_{drag} \Rightarrow \Re_{in}$$

Analogously, this kind of relationship allows us to describe different stages of object behaviour. For example, if we want to describe a temporal sequence of behaviour we can do so using induced relationships. Thus, a temporal sequence of object movement with respect to another object can be defined as follows:

$$(\Re_{out} \Rightarrow \Re_{in}) \Rightarrow \Re_{out}$$

This notation allows us to describe dynamic behaviour according to the object evolution. In the next section, we will focus on the spatial system specification using this formalism.

3. Examples

The spatial specification of a system is focused upon the following concepts. Firstly, the objects of the application must be identified and the degrees of freedom for their manipulation (applicable gestures). After that, (topological) spatial relationships must be recognised and associated to system actions.

3.1. File System

In a file system several objects can be identified, such as, for example, folders (representing containers), files (representing documents) or devices (printer). Any object has the following features:

$$\text{icon} = \{ p \in \text{domain(folder, icon, device)}: \ p \text{ satisfies: } \Re_{drag} \}$$

In this system we can identify several relationships attached to system actions.

Delete
$\Re_{del}(p) ::= \exists p \in dom(icon \cup folder) \ \exists t = trash(folder) / (p \sim t) \wedge \Re_{drag}(p)$

A copy gesture is described by the following relationship. An element is dragged to a new folder (it was not inside it before the dragging gesture)

Copy
$\Re_{copy}(p,t) ::= \exists p \in dom(icon \cup folder), \exists t \in dom(folder) / \neg(p \sim t) \Rightarrow (p \sim t) \wedge \Re_{drag}(p)$

The print action is given by the following rule:

Print
$\Re_{print}(p)::= \exists p \in dom(icon), \exists d=printer(device)$ / $(p \sim d) \wedge \Re_{drag}(p)$

Some considerations derive from this example:

- Any direct manipulation action is given by a drag and drop $\Re_{drag}$ gesture by the user.
- Any action in the system is given by a topological relationship based on inclusion tests.
- The semantic of each action depends upon the kind of object which is dragged or the underlying object in which it is dropped. For example the difference between copy and delete is that the target object is a special folder labelled as trash.

3.2. The Series Games

Another interesting example is a children's game because the direct manipulation of objects is part of their learning process. The knowledge is given by rules (ordering, associations, etc.). In a children's game, objects can be moved freely and we only have to specify the relationships among the objects.

This is an easy game which allows us to analyse the child's capacity to make logical sequences (ordering numbers, sizes, shapes, etc) The rules for the game are that the next piece just to the right must have a higher property than the left-hand piece. Pieces can only be placed on the left or right of the first.

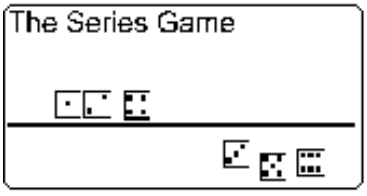

Figure 2. *The Game*

Pieces = { $p \in$ domain(pieces):
p satisfies: $\Re_{drag}$, $\Re_{successor}$ }

The *successor* relationship gives us information about the ordering of the pieces:

$$\Re_{successor}: piece \times piece \rightarrow Boolean$$

where $\Re_{sucesor}(p, q)$ is the satisfaction of the property $p \in$ dom(piece), has less value than $q \in$ dom(piece), and consequently, q is successor of p. The rules are given by the following relationships:

Put on the right
$\Re_{right}(p,q) ::= \exists q \in domain(pieces)$ / $((p \rightarrow q) \wedge (\Re_{successor}(q,p) \wedge \Re_{drag}(p)))$

Placing a piece on the right means that it preserves the partial ordering and such a property is verified by a dragging gesture. Analogously it can be defined "*put on the left*".

Put on the left
$\Re_{left}(p,q) ::= \exists q \in domain(pieces)$ / $((p \leftarrow q) \wedge (\Re_{successor}(p,q) \wedge \Re_{drag}(p)))$

We can restrict the set of valid relationships as follows:

Illegal relationships
$\Re_{invalid}(p) ::= \forall q \in domain(pieces)$ / $(p \uparrow q) \vee (p \downarrow q) \vee (p \sim q)$

4. Conclusions

We have focused on the interaction process based on direct manipulation and have proposed an alternative method based on topological relationships for its specification. Currently, we have a prototype that translates relationships into Tcl/Tk code [7]. Future research should be oriented towards the extension of topological relationships to 3D space, because spatial specification is well suited for application in virtual reality environments.

References

1. M. Harrison, H. Thimbleby (eds.) "*Formal Methods in Human-Computer Interaction*". Cambridge University Press, 1990.
2. B. Shneiderman: "*Designing the user Interface: strategies for effective human-computer interaction, 2ª ed.*". Addison Wesley, 1992.
3. D. Goldberg, A. Gooddisman*: "Stylus User Interfaces for Manipulating Text*". In "*Readings in Human-Computer Interaction: towards the year 2000, 2ª ed.*" Morgan Kaufman Published. 1995.
4. S. Ellis: "*Nature and Origins of Virtual Environments*". In "*Readings in Human-Computer Interaction: towards the year 2000, 2ª ed.*" Morgan Kaufman Published. 1995.
5. Foley, van Dam:*"Computer and Graphics, 2ª ed"* Addison Wesley, 1990
6. H. R. Hartson, K. Mayo: "*A framework for precise, reusable task abstraction*". Design, Specification and Verification of Interactive System (DSVIS'96). Ed. P. Palanque, R. Batisde. Springer Verlag EG, 1996.
7. H. M. Kamal: "*Desarrollo de una librería de Manipulación Directa de Objetos*". Dpt. Lenguajes y Sistemas Informáticos. Universidad de Granada, 1998.