Part III
Chapter 5

Specialized Interactions

5.1 Introduction

Interactions, introduced in chapter 3, were described as constrained actions carried out by two or more cooperating objects. Interactions are the only evidence of activity in a system modelled with objects. The interactions that can take place on a connection are specified in terms of the sets of possible symbols that the connection can convey. Objects, introduced in chapter 4, place constraints on what can happen and different kinds of objects place different constraints on the symbols that are allowed across their interfaces. The interactions described in this chapter arise as a result of connecting different specific kinds of objects together.

A situation in which one object can influence another, but where the reverse is not true results in a particular kind of interaction, called an **announcement**. The affected object has a directed interface.

One object is said to perform an **operation** on another object when the constraint that it imposes on one part of the interaction affects the constraint that the other object can impose on another part of the interaction.

Some interactions can be divided into two parts, that are causally related. This type of interaction is referred to as a **call and reply**. One object, the calling object, issues an announcement to another, the called object. The called object then issues a related announcement back to the original calling object. A **nested call** takes place where a called object, calls a third object to help establish the relation between the two parts of the initial call. A **back-call** is a nested call, where the nested call is made back to the original caller.

The idea of a **non-atomic action** is based on the knowledge that there exists a set of conditions that may influence the parts of an interaction in different ways. All interactions discussed in the context of this manual are assumed atomic. That is, it is assumed that the conditions that apply to the determination of interactions are stable. Even if they could change they are deemed insignificant for the purpose of the model in hand. This assumption is necessary for the design to proceed. When the assumption is invalid however, there is a need to explicitly model the conditions and the objects that impose them.

5.2 Reference section

In the following manual pages the concepts of announcement, operation, c
back-call, nested call, and atomic interaction are described.

NAME

Announcement

PURPOSE

To allow one object to constrain the activity of another.

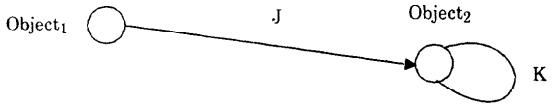
SYNOPSIS

An announcement is an interaction between two objects across a directed interface. One object provides a constraint on another object through a directed interface. The object with the directed interface cannot impose any constraints on the symbols employed in the interaction.

CANONICAL FORM

An announcement involves two objects with a connection between them as shown in Figure 5.1. $Object_2$ is a causal object that engages in activities that can be observed on the connection labelled K. The details of the causal relation it maintains between J and K are not relevant to the definition of an announcement. The interaction between $Object_1$ and $Object_2$ is called an announcement.

Figure 5.1 An announcement from $Object_1$ to $Object_2$



SPECIFICATION

The objects in Figure 5.1 interact using alphabet J. Suppose that $Object_1$ places a constraint J_1 on the symbols that are allowed on the connection. Because of the directed interface of $Object_2$ it cannot place any constraints on the symbols allowed on the connection. The set of possible announcements between $Object_1$ and $Object_2$ is determined only by $Object_1$ and defined by:

$J_1 \subseteq J$

A particular announcement is represented by a lower case letter j, and is bounded by the two constraints J and J_1. That is

$j \in J \wedge j \in J_1 \Leftrightarrow$

$j \in J \cap J_1 \Leftrightarrow$

$j \in J_1$

The announcement may be determined, indetermined or result conflict, depending on the cardinality of the set J_1, from which actual announcement will be taken.

Interactions in alphabet K represent activity in $Object_2$ and v generally be affected by the announcement according to the relat maintained by $Object_2$ between alphabets J and K.

REALIZABILITY ISSUES

SEE ALSO

INTERACTION
DETERMINED INTERACTION
INDETERMINED INTERACTION
UNDETERMINED OBJECT
CAUSAL OBJECT

NAME

Operation

PURPOSE

To allow one part of an interaction to affect the constraints an object imposes on the other parts of an interaction.

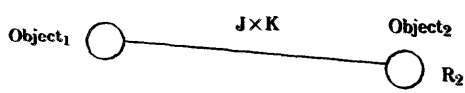
SYNOPSIS

An operation is an interaction that can be decomposed into two parts. One part, often called the operation name with possible arguments, plays a role in determining the constraints that are applied by an object on the other parts of the interaction. Often operations are formed of sequentially composed symbols where the early parts of the symbol help select the constraints imposed later on.

CANONICAL FORM

An operation involves two objects, as shown in figure 5.2. The alphabet conveyed by the connection between them is split into two parts. At least one of the objects maintains a dependency relation between the two parts of the interaction. In figure 5.2 $Object_2$ maintains the relation R_2.

Figure 5.2 An operation involves a decomposed interaction.



SPECIFICATION

Suppose, as shown in Figure 5.2, the alphabet of interaction is given by the composition J × K and $Object_2$ maintains the relation R_2 between the two parts. From the perspective of $Object_2$ the set of possible interactions is taken from the set of pairs of symbols that satisfy the relation R_2. A valid interaction, (j, k), satisfies the predicate

$(j, k) \in R_2$

This constraint can be expressed in a different form. Assume that a symbol j is conveyed on the connection to represent one part of the

interaction where $j \in J$. The constraint on K, imposed by $Object_2$, then restricted both by the relation, R_2, that it maintains and by t symbol j. The actual constraint imposed by $Object_2$ depends on t symbol j, chosen from J. Since there are many possible symbols in the constraint imposed by $Object_2$ on K can be written as a set I where K' is dependent on j, the selected element from J. K' is said be a function of j:

$$K'j = \{k \mid (j, k) \in R_2\}$$

In general different choices for the symbol j create differe constraints on the symbol selected from K. Expressed in this wa appears to combine with the constraint imposed by $Object_2$ on th part of the interaction taken from the alphabet K. The parti interaction j appears to change $Object_2$ and the constraints applies. $Object_2$ is said to be operated upon by the parti interaction j, and the symbol $j \in J$ is called an operation name. T alphabet J is then considered to be an alphabet of operation nam where each name is associated with a potentially differe constraint, imposed by $Object_2$.

Often operations are a part of sequenced interactions where t constraints can be written in the form

$$K'j = \{k \mid (j \rightarrow k) \in R_2\}$$

In a sequenced interaction, the symbol j is determined first and th symbol k is selected. Once the objects have begun to determine t symbol taken from the alphabet K there is no opportunity to chan the symbol selected from J. In this way the constraint K'j is seen be causally related to the symbol j.

REALIZABILITY ISSUES

The operation name is often structured. A part of the name may fixed, another part may be variable. The variable part is oft referred to as the arguments of the operation. The distincti between the fixed and variable part is an arbitrary one, often ma for the convenience of the designer. The parts in the structur operation name may also be used to govern the routing of operation through a network of nodes.

If the two parts of an operation are causally related, then t operation name must precede the remainder, otherwise t operation name cannot be said to determine the constraint on t remainder of the interaction.

NAME

Call and reply

PURPOSE

To allow one object to causally constrain the causal constraint imposed on it by another object.

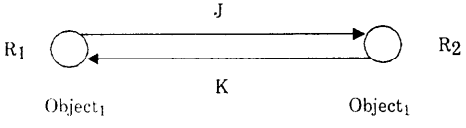
SYNOPSIS

A call and a reply are parts of an interaction between two objects. One of the objects, the called object, maintains a relation which introduces a dependency between the call and the reply. The other object, the calling object, treats the two parts as independent. The calling object determines the call but cannot (directly) determine the reply.

CANONICAL FORM

A call and reply involve two objects. Each object has two interfaces, one of which is directed. The objects share connections as illustrated in Figure 5.3. The connection labelled with the alphabet J is used for the call. The connection labelled with the alphabet K is used for the reply.

Figure 5.3 Call and reply require two connections between two objects



SPECIFICATION

In figure 5.3 $Object_1$ determines the element from alphabet J by imposing a constraint J_1. Because of the directed interface $Object_1$ must allow any element of alphabet K. The relation $Object_1$ maintains is

$R_1 = J_1 \times K$

$Object_2$ cannot place any constraint on the choice of J but does maintain the causal relation R_2 to generate an element of K:

$R_2 \subset J \rightarrow K$

Possible interactions between $Object_1$ and $Object_2$ are given by intersection of the constraints applied by both objects,

$R_1 \cap R_2 \subset (J_1 \times K) \cap (J \rightarrow K)$

which can be written as

$R_1 \cap R_2 \subset J_1 \rightarrow K.$

REALIZABILITY ISSUES

The nature of the dependencies and the directed interfaces enf an order on the formation of symbols.

The underlying mechanism of $Object_1$ is responsible for constraints that determine which interaction is to take place selecting the element of alphabet J used in a particular call. underlying mechanism of $Object_2$ is responsible for the constra that represent the relation to be maintained.

NAME

Back-call

PURPOSE

A back-call divides the task of maintaining a relation between the calling and the called objects.

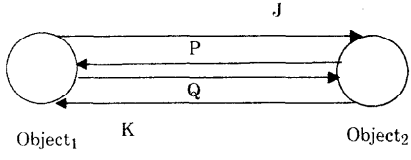
SYNOPSIS

In a back call, an object responding to a call employs the calling object to help maintain the relation anticipated by the original call.

CANONICAL FORM

A back-call involves two objects. Each object has four interfaces, two of which are directed. The objects share connections as illustrated in Figure 5.4. The connections labelled J and K are associated with the call and return. The connections labelled P and Q are associated with the back-call and its return.

Figure 5.4 A back call requires four connections between two objects



SPECIFICATION

Referring to Figure 5.4, $Object_1$ determines the interaction taken from the alphabet J. Assume that this involves imposing constraint J_1. $Object_1$ also maintains a causal relation between alphabets P, Q and J but cannot impose any restrictions on the alphabets P and K. The relation maintained by $Object_1$ can be written as

$R_{1jpqk} \subset J_1 \times P \rightarrow Q \times K$

$Object_2$ maintains a relation between all the alphabets but cannot restrict the choice of alphabets J and Q. The relation maintained by $Object_2$ can be written as

$R_{2jpqk} \subset (J \rightarrow P) \times (Q \rightarrow K)$

The resulting interactions are taken from the set given by the intersection of the constraints imposed by both objects

$R_{1jpqk} \cap R_{2jpqk} \subset J_1 \rightarrow P \rightarrow Q \rightarrow K$

REALIZABILITY ISSUES

The underlying mechanism of $Object_1$ is responsible for the constraints that determine which particular call is to take place by selecting the element of alphabet J. The underlying mechanism of $Object_2$ together with the mechanisms of $Object_1$ are responsible for the constraints that together form the relation and result in the selected interaction.

SEE ALSO

CALL
NESTED CALL

NAME

Nested call

PURPOSE

A nested call allows the relation, that is to be maintained by a call, to be distributed between two objects.

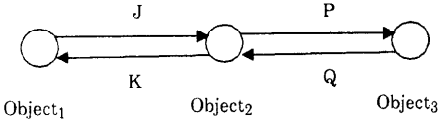
SYNOPSIS

A called object that is to respond to a call can itself employ a call to a third object. The original called object and the additional called object together maintain the relation anticipated by the original calling object.

CANONICAL FORM

A nested call involves three objects. The objects share connections as illustrated in Figure 5.5. The directed connections show that $Object_1$ is responsible for selecting members of the alphabet J, $Object_2$ is responsible for selecting members of the alphabet P and K, and $Object_3$ is responsible for selecting members of the alphabet Q.

Figure 5.5 A nested call involves three objects



SPECIFICATION

$Object_1$ is responsible for the constraint on the alphabet J but treats members of the alphabet J independently of members of the alphabet K. The relation maintained by $Object_1$ can therefore be written as

$J_1 \times K$

This includes the assumption that $Object_1$ imposes the constraint J_1.

$Object_2$ maintains a relation between J, Q, P and K,

$R_{2jqpk} \subset (J \times Q) \rightarrow (P \times K)$

and $Object_3$ maintains the causal relation

$R_{3pq} \subset P \rightarrow Q$

between the members of the alphabets P and Q.

In combination $Object_2$ and $Object_3$ maintain a relation between and K

$R_{23jk} = \{(j \rightarrow k) : \exists p . \exists q . ((p \rightarrow q) \in R_{3qp} \wedge ((j, q) \rightarrow (p, k)) \in R_{2jqpk}) \}$

as though it was maintained by a single object.

REALIZABILITY ISSUES

The underlying mechanism of $Object_1$ is responsible for th constraints that determine which interaction is to take place b selecting the element of alphabet J used in a particular call. Th underlying mechanisms of $Object_2$ and $Object_3$ are responsible fc the constraints that represent the relation that is to be maintaine between J and K.

SEE ALSO

CALL

FUTURE DIRECTIONS

A called object can make use of several further objects, to fulfill it obligation with respect to the calling object. The specification wi need to be extended to allow this. For now however, it is best to loo upon such a situation as if it were a composition of the simple cas presented here.

NAME

Non-atomic action

PURPOSE

To represent interactions between objects which are subject to changing conditions.

SYNOPSIS

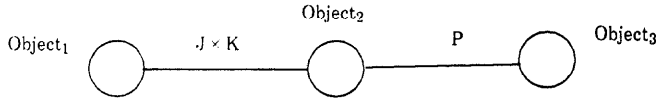
The concept of a non-atomic action is based on the idea that there exists a set of conditions that influence an interaction. Assume an interaction is decomposed into two or more parts. The conditions that apply to the interaction as a whole do not remain steady during the determination of its parts. The interaction is then referred to as a non-atomic action.

All interactions discussed in the context of this manual are assumed atomic. That is, the conditions that apply to the determination of the interactions are usually assumed stable, and when not stable they are deemed insignificant to the outcome. This assumption is necessary to allow the design to proceed. When it is necessary to describe what happens when this assumption is invalid, there is a need to explicitly model the conditions and the objects that impose them.

CANONICAL FORM

Figure 5.6 shows two objects, $Object_1$ and $Object_2$, interacting through a composite interaction in $J \times K$. $Object_2$ is also interacting with $Object_3$ using the alphabet P. The members of the alphabet P are interpreted as the conditions under which $Object_2$ enters into the interactions with $Object_1$.

Figure 5.6 $Object_3$ places conditions on $Object_2$ and so on the interactions between $Object_1$ and $Object_2$



SPECIFICATION

From the perspective of $Object_1$ in Figure 5.6 there is a require relation

$$R \subseteq P \times J \times K$$

to be maintained by $Object_2$. This required relation indicates that i the interactions between $Object_1$ and $Object_2$ are of the form (j, k) then the interactions are dependent on the condition taken from P The set of required interactions is given by

$$\{(j, k) \mid (p, j, k) \in R\}$$

where p, the condition, is written here as a free variable but it is t be determined by some part of the system.

The object responsible for maintaining the relation, $Object_2$, can b decomposed into two parts as shown in figure 5.7. Each part i subject to potentially different conditions. These two parts can b considered either to represent two spatially separated mechanisms responsible for maintaining the same constraints, or to represen the same mechanism at two different times. One part, labelle $Object_{2a}$, provides directly a constraint on the alphabet J and th other part, labelled $Object_{2b}$, provides directly a constraint on th alphabet K. $Object_{2a}$ is subject to a condition from the alphabet P while $Object_{2b}$ is subject to a condition conveyed on the connectio labelled P_b. Because the conditions in P_a are different from those i P_b, the sets of conditions are disjoint

$$P_a \cap P_b = \{\}$$

The connection between the two parts of $Object_2$ carries symbol from an alphabet I. Alphabet I is normally considered as part of th internal activities of $Object_2$ and ensures that if the two parts o $Object_2$ are dependent upon one another interactions to that effec can take place in the decomposed system.

$Object_{2a}$ maintains a relation

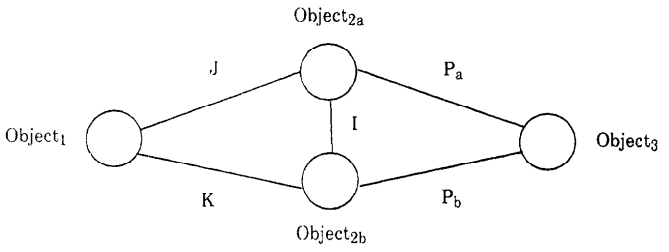
$$R_{2a} \subset P_a \times J \times I$$

and $Object_{2b}$ maintains the relation

$$R_{2b} \subset P_b \times K \times I$$

The relation maintained by $Object_2$ can be derived from the tw relations R_{2a} and R_{2b}

Figure 5.7 The two parts of Object$_2$ are subject to potentially different conditions



$R_{jk} = \{(j, k) | \exists i . ((p_a, j, i) \in R_{2a} \wedge (p_b, i, k) \in R_{2b})\}$

which is equivalent to R when the conditions p_a and p_b are the same. The relations R_{2a} and R_{2b} must therefore be chosen so that when $p_a = p_b = p$

$R_{jk} = \{(j, k) | \exists i . ((p, j, i) \in R_{2a} \wedge (p, i, k) \in R_{2b})\}$

$= \{(j, k) | (p, j, k) \in R\}$

Because the same relation is defined, the predicates that occur in both expressions must yield the same result, which implies that

$\exists i . ((p, j, i) \in R_{2a} \wedge (p, i, k) \in R_{2b}) = (p, j, k) \in R$

and it can then be shown that for any values of p_a and p_b

$R_{jk} = \{(j, k) | (p_a, j, k) \in R \wedge (p_b, j, k) \in R\}$

or

$R_{jk} = \{(j, k) | (p_a, j, k) \in R\} \cap \{(j, k) | (p_b, j, k) \in R\}$

If the condition is not maintained so that $p_a \neq p_b$ the outcome depends on the nature of the relation R and the precise conditions that pertain. It is possible that, with different conditions applied, the result will be a conflict.

REALIZABILITY ISSUES

The possibility of conflict or inconsistency arises where a relation is maintained by separated objects. The separation may be in time or in location.

SEE ALSO

CONFLICT

FUTURE DIRECTIONS

The specification will be extended to expose all steps necessary derive R_{jk}.

5.3 Examples

5.3.1 Announcement

Announcements at the railway station
At railway stations announcements are frequently made. The station master announces the arrival of the trains and informs waiting passengers of the destination. The passengers cannot respond to the announcements. They do however change their activities. When a train arrives that will take me to my destination I will be prompted by the station masters announcement, stop reading my paper, and enter the train. When an announcement is made to say that my train will be late, I may get angry, but I will not affect the station master in any way.

5.3.2 Operation

Storage
There are two parts to the storage interaction: the *write* operation and one *read* operation or a succession of *read* operations. The *write* operation involves a symbol identifying the operation accompanied by a symbol representing data. This might be written as

"Write"$\times$ $data_w$

where $data_w$ is an alphabet of possibilities.

The *read* involves a symbol identifying the action to be taken (to distinguish a *read* from a *write*) and an element from a similar data alphabet. This is written as

"Read"$\times$ $data_r$

A whole interaction involving one *read* can then be written as

"Write"$\times$ $data_w$$\times$"Read"$\times$ $data_r$

The storage user is responsible for selecting the element of $data_w$ and the storage provider maintains a relationship between $data_w$ and $data_r$. The relationship is a straightforward "one-to-one" relationship. Suppose

$data_w = \{0,1\}$

and

$data_r = \{0,1\}$

then the relation between them is

{(0,0), (1,1)}

which ensures that what is read is the same as what is written.

5.3.3 Call

library routines

A program can be written to call a library routine that will perform the cosine function. The library routine maintains a relation between an angle in radians (a floating point number taken from the set of floating point numbers between 0 and 2π, modulo 2π) and a floating point number taken from the set of floating point numbers between +1 and -1 inclusive. The program determines the interaction by choosing a particular angle in radians in the call. The library routine performs the function and produces the other part of the interaction.