Part III
Chapter 8

Arrangements of Objects

8.1 Introduction

Objects can be arranged in may ways. There are however several forms in which objects are often arranged and these patterns are described in this chapter.

Objects are often arranged in a so called **hierarchy**. In a hierarchy some objects are constrained by others but are unable to impose constraints themselves. In situations where objects are considered of equal status or seniority, it would be inappropriate to connect objects in a hierarchy. An arrangement in which the objects are bound such that they may impose constraints on one another to ensure the achievement of a common goal is called a **federation**.

In the **client-server** pair, the objects are called the client and the server. The client retains control over the server.

The **peer-to-peer** relationship is symmetrical and both related objects act as equal partners.

8.2 Reference section

In the following manual pages the concepts of federation, hierarchy, client-server pair and the peer-to-peer arrangement are described.

NAME

Federation

PURPOSE

A federation is an arrangement of objects where the objects all the mutual imposition of constraints to assist in achieving sor common goal.

SYNOPSIS

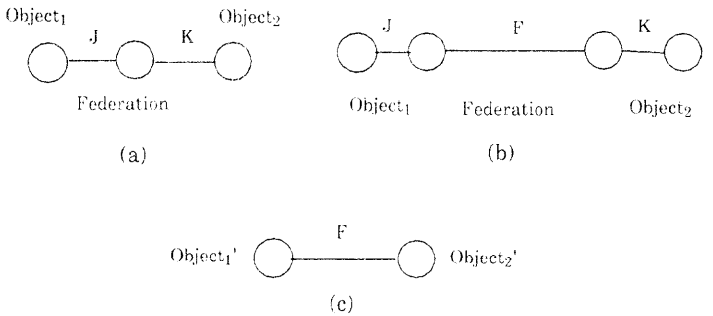
A federation has connotations of cooperation amongst the federat parties. Cooperation, where there is a potential conflict, involv negotiations which lead to an agreement. The agreeme constrains the activities of the partners and assists them both achieving their goals. The negotiated constraints are intended avoid attempts at interactions that result in conflicts of interest.

In a federation all objects or collections of objects are able to impc constraints on all other objects or collections of objects.

CANONICAL FORM

The canonical form for two federated objects is shown in figure .1(a) and involves the two federated objects that are connected the federation relationship. The details of the relationship depe upon the nature of the agreement between the partners.

Figure 8.1 A federation

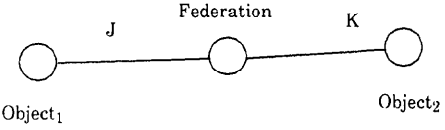


A federation relationship allows each object to interact with all of the others in the federation and in the case of two objects can be decomposed as shown in figure 8.1(b). There is no characteristic structure in a federation and the links between the two parts are shown simply as a connection carrying a separate alphabet, F. The two parts of the relationship can be subsumed into the two objects as shown in figure 8.1(c). This makes each object responsible for administering a part of the federation relationship.

SPECIFICATION

Figure 8.2 includes two federated objects. The activity relevant to the federation and taking place in $Object_1$ is indicated by symbols from the alphabet J. Evidence of the activity in $Object_2$ which is relevant to the federation appears on the connection labelled K. The objects impose constraints on one another through the federation which is a relationship that has been established after some negotiation.

Figure 8.2 A federation of objects.



The constraints on activity in the two objects can be considered in two parts: the internal constraints which limits activity when federation is not considered and the mutual constraint which is a result of federation. The internal constraint imposed by $Object_1$ on the interaction J can be written as

$J_1 \subseteq J$

and the internal constraint on $Object_2$ can be written as

$K_1 \subseteq K$.

The activity within the two objects is represented by pairs of symbols of the form (j, k) which is subject to the internal constraints. The combined internal constraints can be written as

$J_1 \times K_1$

which identifies the bounds of activity without federation.

The activity within the two objects is also constrained through t federation. Assuming the relation between the interactions $Object_1$ and $Object_2$ maintained by the federation is R_f then t combined constraint is

$R_f \cap J_1 \times K_1$

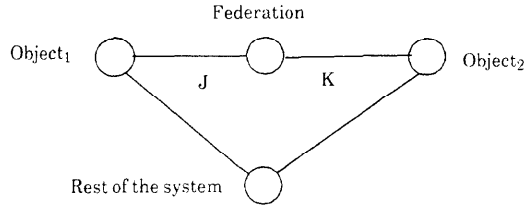
and activities in the two objects will be bounded by this relation.

REALIZABILITY ISSUES

It is possible that the additional constraints imposed from t federation lead to a system that is overconstrained and activ becomes impossible. Additional objects that incorporate techniq for the prediction of conflict and its avoidance may be needed.

The constraints on activities that objects place on one another m be derived from being a part of a federation but they may also co from other relationships established through the rest of the syst as shown in figure 8.3. This can be accommodated in the form model by elaborating on the internal constraints imposed by t objects in the federation.

Figure 8.3 A federation as part of the relationship between two objects

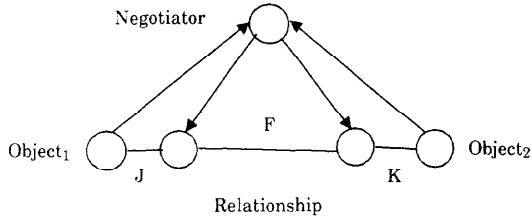


The federation relationship must be negotiated and in a comple model of the system any negotiator and the partners in t negotiation should be included. Figure 8.4 illustrates one way which this might be done.

SEE ALSO

TRADERS

Figure 8.4 An illustration of how a negotiator might help establish a federation



FUTURE DIRECTIONS

The entry must be seen as a place holder. The text will be reviewed and aligned with entries for trader and negotiation.

NAME

Hierarchy

PURPOSE

A hierarchy is an arrangement of objects that allows some objects t
impose constraints on others.

SYNOPSIS

In a hierarchy one object or collection of objects is able to impos
constraints on another object or collection of objects. In a hierarch
reciprocal constraints cannot be applied. It may be argued that
true hierarchy does not and cannot exist.

CANONICAL FORM

The canonical form is shown in figure 8.5(a) involving two objec
connected through a hierarchical relationship. A decomposition
the relationship reveals, in figure 8.5(b), that the relationshi
includes a directed interface. Because of the directed interface
$Object_2$, $Object_1$ is able to impose constraints on activity in Object
$Object_2$ is unable to affect activity in $Object_1$. The relationship ca
be subsumed into the interacting objects as shown in figure 8.5(
where $Object_2'$ is shown with a directed interface.

By convention, when convenient to do so, the object that imposes th
constraints is drawn above the object that is subject to th
constraints.

SPECIFICATION

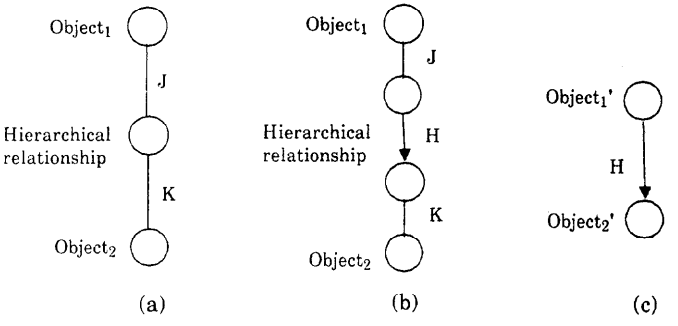
To illustrate the effect of a hierarchy the lower of the two objects i
decomposed to reveal evidence of its activity as shown in Figure
.5(b). $Object_1$ imposes constraints on the activity of $Object_2$ throug
the hierarchy relationship. The activity in $Object_1$ is indicated b
symbols on the connection labelled J and the activity in $Object_2$ i
indicated by symbols from the alphabet K. $Object_1$ imposes
constraint on the alphabet J. Assuming the constraint imposed o
alphabet J is

$J_1 \subseteq J$

and that $Object_2$ imposes a constraint on the symbols taken from th
alphabet K

$K_1 \subseteq K$

Figure 8.5 A hierarchy of objects



Assuming that overall the hierarchy relationship imposes a relation R_{jh} between J and H such that

$R_{jh} \subseteq J \times H$

and a relation R_{hk} between the alphabet H and the alphabet K so that

$R_{hk} \subseteq H \rightarrow K$

then the set of interactions that can take place involving alphabets J, K and H is given by

$(J_1 \times R_{hk}) \cap (R_{jk} \times K_1)$

$$\subseteq (J_1 \times H \rightarrow K) \cap (J \times H \times K_1) = (J_1 \times H \rightarrow K_1)$$

and the set of possible interactions on the connection labelled K is given by

$\{k \mid \exists j \,.\, \exists h \,.\, (j, h) \in R_{jh} \wedge (h \rightarrow k) \in R_{hk} \wedge j \in J_1 \wedge k \in K_1\}$

It is possible that this set is empty, which indicates that there is a conflict and no interaction is possible on the connection labelled K. $Object_1$ can overconstrain activities in $Object_2$.

REALIZABILITY ISSUES

It is possible that the additional constraints imposed from the hierarchy lead to a system that is overconstrained and activity

becomes impossible. Techniques for the prediction of conflict and i resolution may need to be added.

For a strict hierarchy to be maintained there should be ı possibility of the objects interacting with the rest of the system anything but a hierarchical fashion. It is clear that su restrictions cannot be enforced in a physical system. A model such a system that includes a hierarchy may still be a valid model the reverse constraints that are unavoidably imposed a insignificant for the purpose of the model.

SEE ALSO

INTERACTION
CAUSAL OBJECT
CONFLICT

FUTURE DIRECTIONS

The specification will be reviewed. It is expected that this will le to a more compact description.

NAME

Client-server pair.

PURPOSE

To maintain a relationship that allows the client to retain control over the server.

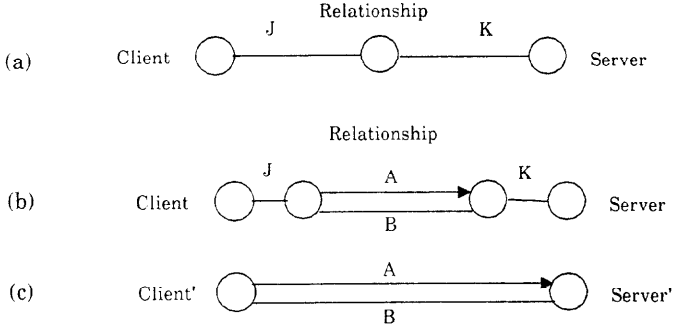
SYNOPSIS

A client server pair is an asymmetrical arrangement which allows the client to determine the nature of the interaction. The arrangement may be expressed as an object or relationship, sometimes referred to as the client-server relationship.

CANONICAL FORM

A client-server pair may be represented by an arrangement of three objects: the client, the server, and the relationship object. The relationship, shown in Figure 8.6(a), can be decomposed to reveal its asymmetrical structure. Figure 8.6(b) shows that the relationship can be split into two parts, one of which is directional. When the relationship is subsumed into the objects that it relates, as in Figure 8.6(c), the directed and undirected interfaces remain.

Figure 8.6 A client-server relationship



SPECIFICATION

In Figure 8.6(b) the activity in the client is indicated by symbols c the connection labelled J and the activity in the server is indicat by symbols from the alphabet K. The client imposes a constraint c the alphabet J. Assuming the constraint imposed on alphabet J is

$J_1 \subseteq J$

The server imposes a constraint on the symbols taken from tl alphabet K

$K_1 \subseteq K$

Assuming that overall the client-server relationship imposes relation R_{jab} between J and A and B such that

$R_{jab} \subseteq J \times A \times B$

and a relation R_{abk} between the alphabet A, B and the alphabet so that

$R_{abk} \subseteq A \rightarrow (B \times K)$

then the set of possible interactions for the client-serv relationship is given by

$$(J_1 \times R_{abk}) \cap (R_{jab} \times K_1)$$
$$\subseteq (J_1 \times A \rightarrow (B \times K)) \cap (J \times A \times B \times K_1)$$
$$= (J_1 \times A) \rightarrow (B \times K_1)$$

Expressed in terms of the alphabets used by the original client ar server the interactions can be written as

$$\{(j \rightarrow k) \mid \exists a \,.\, \exists b \,.\, (j, a, b) \in R_{jab} \wedge (a \rightarrow (b, k)) \in R_{abk} \wedge j \in J_1 \wedge k \in K_1\}$$

REALIZABILITY ISSUES

SEE ALSO

OBJECT WITH A DIRECTED INTERFACE
CAUSAL OBJECT

FUTURE DIRECTIONS

NAME

Peer-to-peer arrangement

PURPOSE

To allow two objects with equal status to interact.

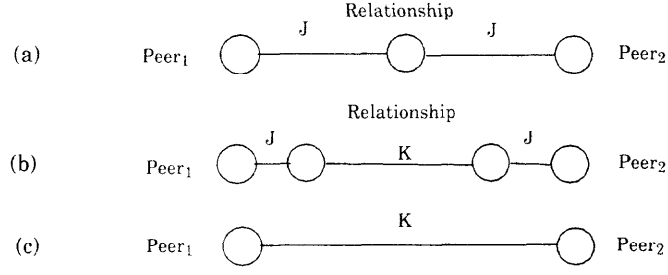
SYNOPSIS

A peer-to-peer arrangement, expresses the idea that two objects may interact as equal partners. It is a symmetrical relationship where each object can constrain the activities of the other.

CANONICAL FORM

The peer-to-peer arrangement is represented by three objects: the peers and the relationship between them. The relationship in Figure 8.7(a) can be decomposed to reveal, in Figure 8.7(b), that it simply incorporates a connection which can be subsumed into the peer objects by changing their interface specifications as shown in Figure 8.7(c).

Figure 8.7 A peer-to-peer relationship can be represented as a connection



SPECIFICATION

A peer-to-peer relationship can be treated as though it provided a connection for subsets of the activities that take place in the related objects.

REALIZABILITY ISSUES

Although the peer-to-peer relationship can be represented by single connection, there may be some elaborate underlyir technology that ensures that that connection is made.

SEE ALSO

PAIR OF CONNECTED OBJECTS
INTERACTION

FUTURE DIRECTIONS

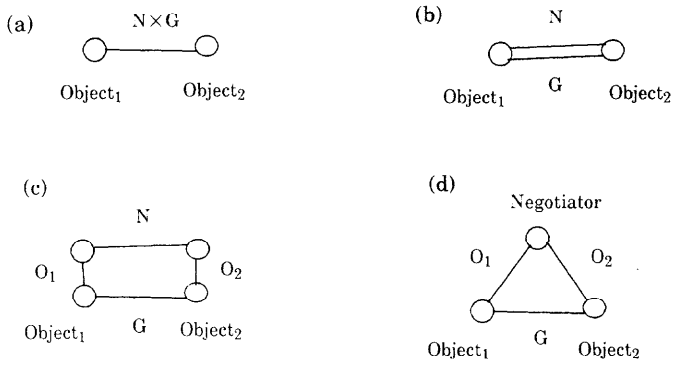
It may be necessary to extend the specification. Currently implicitly refers to Chapter 2.

8.3 Examples

8.3.1 Federation

Negotiators
Federation implies a degree of cooperation. Often federation is considered to consist of two separate interactions: routine interaction and negotiation. Negotiation involves the objects trading with one another to reach a compromise and an agreement on how routine interactions are to take place. The outcome of the negotiation is a convention that places bounds on routine interactions and establishes their form. Figure 8.8(a) shows two federated objects where the alphabet of interaction is split into two parts. N is the part employed solely for negotiation and G is the part which includes all the possible forms of routine interaction that will be constrained by the negotiated convention.

Figure 8.8 Negotiation of conventions is sometimes considered separately.



The separate alphabets can be shown, as in Figure 8.8(b), associated with two separate connections. In Figure 8.8(c) the system is decomposed further by splitting the objects into two parts: one part involved in negotiation and the other engaged in routine interaction. The part involved in negotiation in both objects has to convey the outcome, using alphabets O_1 and O_2, to those parts engaging in the routine interaction. Figure 8.8(d) combines the objects involved in negotiation into a single negotiator or trader.

8.3.2 Hierarchy

Management hierarchy.
In a management hierarchy a management object imposes changes on the objects that it is managing in response to changes in its environment. Note that if the managed objects can communicate or impose constraints on the

management object directly or through the environment then th arrangement is no longer hierarchical.

8.3.3 Client-server pair

services and trading

A server is often viewed as an important and centralized resource that i subject to many demands from many different clients. The asymmetry of th client-server relationship is visible in the interactions that can take plac between them. Most often it is the client that chooses what interaction is t take place. The server is like a slave and yields to the clients demands. Th server can however decide when to make its service available and when t withdraw it. In that situation, the server must announce the availability o otherwise of the services it offers. This announcement is often performe through a third object, the trader.

incomplete design

During the design of a system, it is desirable to be able to reason about part of the system without a knowledge of other parts that have not yet bee designed. An incomplete design will include objects whose specification i incomplete or that interact in part with objects whose specification is not ye known. The lack of knowledge means that the best that can be done is t provide a set of possible constraints and to represent the part of th uncompleted design as a client.

services in unknown environments

A similar situation arises where a designer is designing system component that are to be used in a variety of systems in conjunction with a variety o other components. Some systems in which the components are to be use will not have been conceived at the time the component design is done. Th component can only be specified in terms of the range of constraints that i can impose rather than a particular constraint imposed in a particula situation.

8.3.4 Peer-to-peer arrangement

Protocols in OSI

A protocol in OSI has been defined as the rules of engagement between peers i.e. between entities in the same layer. The rules of engagement betwee entities in different layers are defined by services. In many of the protocol in the OSI stack negotiations take place. At the presentation layer the data representation is the subject of negotiation. At the data-link layer the packet sizes can be negotiated. In the network layer certain routing protocols are the subject of negotiation. Often this negotiation takes place on the exchange of connect protocol data units. The fact that an agreement is reached is symbolized by the successful establishment of a connection. The result of the negotiation stands for the duration of the existence of the connection, but sometimes negotiations can be entered into during a connection. This is the case in certain transport protocols, where the buffe

sizes and frequency of acknowledgements can be adjusted to suit both communicating systems and the amounts of data that flows between them.

When using connectionless protocols a similar negotiating procedure becomes more difficult to implement. The negotiations may have been conducted in a previous epoch, in which case there is no need to repeat them at runtime.

Negotiation at runtime increases flexibility, but reduces the speed at which data can be exchanged.