Part III
Chapter 13

Trading

13.1 Introduction

The concept of trading is an important one in many areas of distributed computing and outside it. The essence is captured by a matching process that can bring two or more parties together so that they may interact without conflict and with reduced indeterminacy.

An **interface adapter** is an object that can transform the symbols that an object uses for its interactions into other symbols, better matched with the environment of that object. Interface adapters need instructions as to how to perform the transformation and use an **interface specification** for this purpose.

Two objects may need to engage in some **negotiations** to ensure that the rest of their interactions can take place without conflict and possibly with reduced indeterminacy. Performing such negotiations is sometimes called trading.

The negotiations can then be conducted through a third object, called a **trader** and the result of the negotiations may cause the interface specifications used by the interface adapters to be modified to achieve the trading aim, namely to eliminate conflict and reduce indeterminacy.

Editorial:
There are several issues surrounding the trading function that have not been made explicit in this chapter. In future releases it is anticipated that these issues will be addressed. They are briefly identified as:
Second party trading. To describe a situation in which one of the interacting objects has subsumed the trading function.
Trading and epochs: The interactions with the trader may make interfaces available (offer) or may request their availability. This may be done at different times. It is also possible that an interface that has been offered is withdrawn.
Trading domains, scope, context. It is often thought that trading is not only about matching the interfaces of two known objects, but also about matching the interface of a known object to an interface of one of a number of other objects in a collection or group. Such a group may be modelled by an undetermined object. In this respect the relation between trading and binding must be made explicit.

Multiple trading domains, federated traders or trading traders. When trading domains have been introduced the idea of hierarchies and federations can be formalized.
Other issues include *trading and groups* and *fourth party trading*, where the exporter is not the object with the interface.

13.2 Reference section

In the manual pages that follow the concepts of interface adapter, interface specification, negotiation and the trader are described.

NAME

Interface adapter

PURPOSE

To convert one form of interaction into another

SYNOPSIS

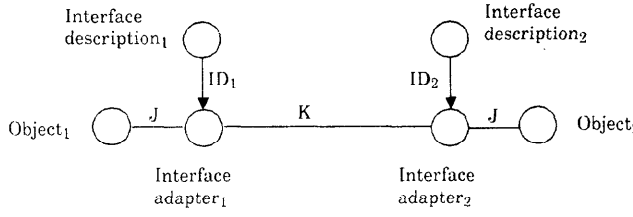
A design will contain several interconnected objects. The designers need to suggest ways in which the system may be implemented. They do not however have complete control over the available components. An interface adapter may be attached to an object so that the interface of that object becomes matched with the available connection technology. The interface adapter needs to convert the specified interactions into a structured form of interactions taken from the actual (available) interconnection alphabet.

CANONICAL FORM

An interface adapter is represented by an object that is interposed between an object and the connection it uses for its interactions. The interface adapter requires an interface description that governs the way in which the interface adapter behaves.

In Figure 13.1 two interacting objects, $Object_1$ and $Object_2$, are shown to each have an interface adapter that is governed by an interface description.

Figure 13.1 Interface adapters



SPECIFICATION

The alphabets J and K in Figure 13.1 do not have to have the same cardinality. If the cardinality of J is larger than that of K then the

interface adapter must ensure that each symbol in J correspon several symbols in a structure in K. If the cardinality of K is la than that of J, then there is a need to make a choice as to whi the possible symbols in K should represent each symbol in J. this the interface adapter requires a policy, which will be part o interface description.

Because there are many ways in which an interaction taken fr can be converted into a possibly structured interaction taken f K, the interface adapter requires some guidance in the form o interface description.

REALIZABILITY ISSUES

The need for an interface adapter stems from realizability is and in particular the availability of components that impleme specification of connected objects.

The interface adapter must be able to interpret the inter description.

SEE ALSO

INTERPRETER
MODEL
POLICY

FUTURE DIRECTIONS

Provide improved specification. The alphabets on the two objec Figure 13.1 do not have to be the same. The relationship binders and binding must be exposed. The interface specificat are really denotations and the interface adapters are spe purpose interpreter.

NAME

Interface description

PURPOSE

To govern the way in which an interface adapter converts one form of interaction into another.

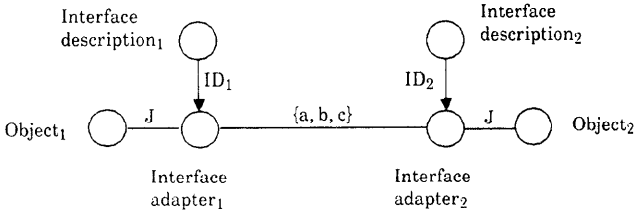
SYNOPSIS

Interface adapters may be used to convert one form of interaction into another. Because there are many ways in which an interaction can be converted into another form, there is a need for an interface description that specifies the required forms of interaction without actually providing them.

CANONICAL FORM

Figure 13.2 illustrates the canonical form. Interface descriptions are attached to interface adapters. The interface adapters cannot influence the interface descriptions. An interface description governs the relationship maintained by the interface adapter.

Figure 13.2 Interface descriptions govern interface adapters



SPECIFICATION

The interface description may consist of two parts. One part describes how interactions taken from J are converted to structures of interactions taken from K. Another part will specify how possible ambiguous mappings are to be treated. If a symbol in J can be converted to one of several possible symbols or structured symbols in K, then a choice needs to be made. A policy helps making this

choice. A policy may further reduce the potential for confl between the interface adapter and its environment.

REALIZABILITY ISSUES

The interface description must be interpretable by the interf adapter.

SEE ALSO

MODEL
POLICY

FUTURE DIRECTIONS

Provide extended specification. Strengthen the links with mod policy and interpreters.

NAME

Negotiation

PURPOSE

Negotiations are the interactions that are aimed at reaching or confirming an agreement over the conventions that must be applied for the useful interaction between objects.

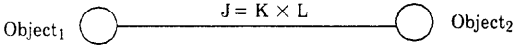
SYNOPSIS

Two objects may devote part of their interactions to negotiations that are aimed at reaching or confirming an agreement over the conventions that must be applied for the useful interaction between them. The negotiation is to reduce the indeterminacy and avoid conflict. The negotiation can take place before or during the other interactions.

CANONICAL FORM

Negotiation between two objects is part of their interaction. The general canonical form of two objects that can negotiate and otherwise interact is shown in Figure 13.3 where the interaction alphabet J, used between two objects, has been split into two parts. One part, called K, is devoted to negotiation. Another part, called L is devoted to interaction within the bounds that have been set by the negotiation.

Figure 13.3 Negotiation as part of an interaction



SPECIFICATION

$Object_1$ and $Object_2$ both maintain a relation between the symbols taken from K and those taken from L:

$R_1 \subseteq K \times L$

$R_2 \subseteq K \times L$

$Object_1$ can impose one of a set of constraints on the symbols tak from L

$\{L_{11}, L_{12}, L_{13}, \ldots L_{1m}\}$ where $L_{11} \subseteq L, L_{12} \subseteq L, L_{13} \subseteq L, \ldots L_{1m} \subseteq$

Possible interactions with $Object_1$ are then taken from the set:

$L_{11} \cup L_{12} \cup L_{13} \cup \ldots \cup L_{1m}$

Similarly, $Object_2$ can impose one of a set of constraints on tl symbols taken from L

$\{L_{21}, L_{22}, L_{23}, \ldots L_{2n}\}$ where $L_{21} \subseteq L, L_{22} \subseteq L, L_{23} \subseteq L, \ldots L_{2n} \subseteq I$

and possible interactions with $Object_2$ are then taken from the set:

$L_{21} \cup L_{22} \cup L_{23} \cup \ldots \cup L_{2n}$

After substitution, the relation maintained by both objects can rewritten as

$R_1 \subseteq K \times (L_{11} \cup L_{12} \cup L_{13} \cup \ldots \cup L_{1m})$ and

$R_2 \subseteq K \times (L_{21} \cup L_{22} \cup L_{23} \cup \ldots \cup L_{2n})$

For both $Object_1$ and $Object_2$ the symbol(s) chosen from tl negotiation alphabet K determine the particular constraint or set constraints on L that the objects impose.

$Object_1$ imposes a constraint $K_1 \in K$ on symbols taken from K. T particular constraint that $Object_1$ imposes on symbols chosen from is represented by L_{1p}, where

$L_{1p} \in (L_{11} \cup L_{12} \cup L_{13} \cup \ldots \cup L_{1m})$

The relation R_1 can be rewritten as

$R_1 \subseteq K_1 \times L_{1p}$

Similarly, $Object_2$ imposes a constraint $K_2 \in K$ on symbols tak from K. The particular constraint that $Object_2$ places on tl symbols chosen from L is represented by L_{2q}, where

$L_{2q} \in (L_{21} \cup L_{22} \cup L_{23} \cup \ldots \cup L_{2n})$

and the relation R_2 can be rewritten as

$R_2 \subseteq K_2 \times L_{2q}$

The particular interaction between $Object_1$ and $Object_2$ is tak from

$R_1 \cap R_2 \qquad \Leftrightarrow$

$K_1 \times L_{1p} \cap K_2 \times L_{2q} \quad \Leftrightarrow$

$(K_1 \cap K_2) \times (L_{1p} \cap L_{2q})$

Assuming that the interaction in alphabet K is determined, that is $\#(K_1 \cap K_2) = 1$, then interactions in L are given by

$l \in L \cap L_{1p} \cap L_{2q}$

This interaction may thus be determined, indetermined or result in conflict, depending on whether
$L \cap L_{1p} \cap L_{2q}$ has one member, has several members or is empty.

REALIZABILITY ISSUES

There can be nothing in the general specification of negotiation (as given above) that says anything about the outcome of negotiations. If the negotiation is to reduce the indeterminacy and avoid conflict in the interactions that take place in L, then the designer needs to make a careful choice, such that the resulting interaction in L will be determined.

If there is indeterminacy in the interactions that constitute the negotiation there may need to be some higher level negotiation to eliminate this. This reflects the recursive nature of the concept of negotiation.

An object may impose "one of a set of constraints" on its interaction alphabet for several reasons. The object may be an undetermined object, that is the designer has not made a choice as to which object shall fill that position in the system. If the object is decomposed into an object together with its interface adapter, it is possible that the particular constraints that the object places on its interactions will depend on the particular interface description that is being used. Again, if that choice has not been made, then the object must be regarded as an undetermined object.

SEE ALSO

UNDETERMINED OBJECT
INTERFACE SPECIFICATION
INTERFACE ADAPTER

FUTURE DIRECTIONS

NAME

Trader

PURPOSE

To provide a separate mechanism that can perform the negotiat function.

SYNOPSIS

A trader is an object that can perform the negotiation functi Negotiation can be necessary to determine the interactions betwe two objects. The trader performs the negotiation that would neec take place between two objects and instructs the interface adapt connected to the objects to interact according to some constrair such that the resulting interactions are determined.

CANONICAL FORM

The system illustrated in Figure 13.4 contains an object label "trader". The trader interacts with two requesters, that each t the trader what are the possible interface descriptions that Obje and $Object_2$ can assume. The trader makes a match and select particular interface description, which is then imposed on t interface adapters that connect $Object_1$ and $Object_2$ together. case the trader can choose several particular interface descriptio the trader needs a policy to resolve the ambiguity. In case there no possible interface description, the trader needs to refer to authority so that the conflict may be resolved.

SPECIFICATION

The symbols in ID_1 and ID_2 are not necessarily the same. T trader is responsible for relating the information about t interfaces of $Object_1$ and $Object_2$ via the connections labelled I and ID_2, such that ID_1 and ID_2 are the same if compared throu some intermediate language:

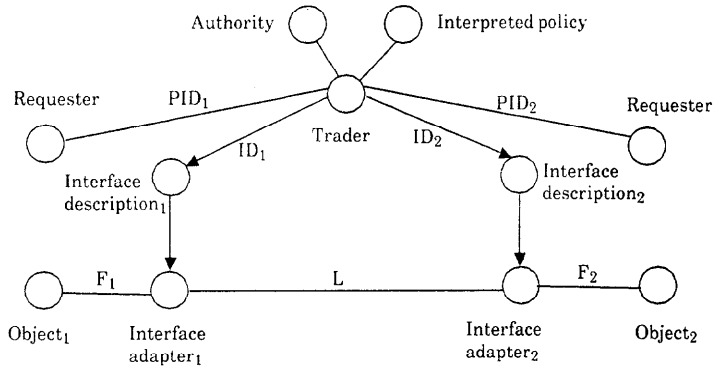
$F_1(ID_1) = F_2(ID_2)$

let X be the set of possible interface descriptions that $Object_1$ a $Object_2$ have in common

$X = F_1(PID_1) \cap F_2(PID_2)$

if the cardinality of X is zero ($\#X = 0$) then help is required, there is no set of interfaces ID_1 and ID_2 through which $Object_1$ a

Figure 13.4 The trader in a system



Object$_2$ can interact. The trader will need to refer to the authority to resolve this conflict.

if $\#X > 0 \Rightarrow F_1(ID_1) \in X \wedge F_2(ID_2) \in X$

The trader may require an interpreted policy to determine which set of interface descriptions to use in case there is a choice, i.e. $\#X > 1$.

REALIZABILITY ISSUES

In the specification above the trader acts on behalf of two objects. In many real systems the trader acts on behalf of a great many objects, matching many requests to many other requests.

SEE ALSO

INTERFACE ADAPTER

FUTURE DIRECTIONS

The specification is to be extended and harmonized with the entry for negotiation.

The assumption that Requester and Object are independent is to be documented.

The application of decomposition on epochs allows us to consider the trading process separately from the interactions between the

objects. The bottom half of Figure 13.4 is then the same as t canonical form for interface adapter and the description does r have to be repeated.

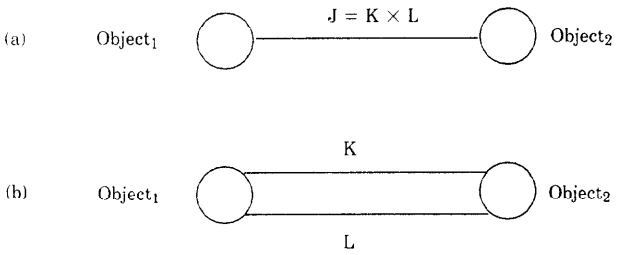
13.3 Examples

13.3.1 Relationship between concepts in this chapter

This section is intended to provide the "cartoon" version of this chapter. It describes how the entries in this chapter can be derived from one another graphically. In particular it concentrates on the transformations that are necessary to turn the canonical form for negotiation into the canonical form for the trader. Many of the transformations used are discussed in detail in Chapters 6 and 18.

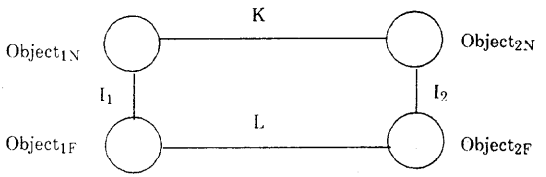
Figure 13.5(a) shows the canonical form for negotiation. The connection between $Object_1$ and $Object_2$ can be decomposed into two parts, as illustrated in Figure 13.5(b). The effect of this is to separate the negotiation alphabet from the rest of the interactions between both objects.

Figure 13.5. Separating negotiation from other interactions



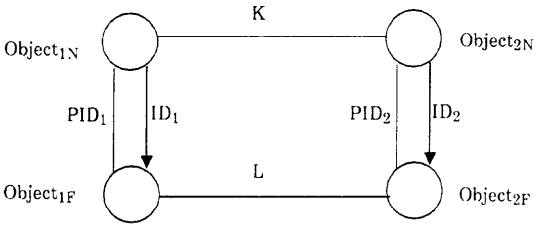
The parts of $Object_1$ and $Object_2$ that are responsible for the negotiation can be separately identified by decomposing the objects into two parts. This is illustrated in Figure 13.6. The parts are not independent and this is shown by the inclusion of an internal alphabet I_1 and I_2 respectively. $Object_{1N}$ and $Object_{2N}$ are responsible for the negotiations, whilst $Object_{1F}$ and $Object_{2F}$ are the functional parts, that are responsible for the rest of the interactions.

Figure 13.6 Isolating the negotiating parts from the objects



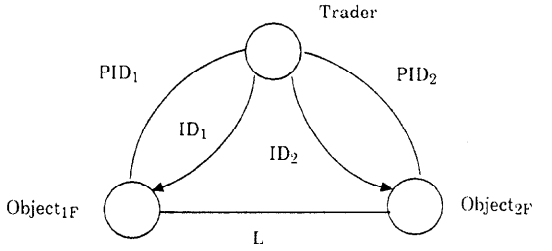
The alphabets I_1 and I_2 are used to determine the particular set of constraints that $Object_{1F}$ and $Object_{2F}$ respectively place upon their interactions with one another. Alphabet I_1 can be refined into two parts. One part, identified by PID_1 in Figure 13.7, is used to signal to $Object_{1N}$ which possible constraints can be imposed. The other part, labelled ID_1 may be used by $Object_{1N}$ to prescribe which particular set of constraints should be chosen. Similarly, the alphabet I_2 may be refined into PID_2 and ID_2. The connections labelled ID_1 and ID_2 are directed because the objects $Object_{1F}$ and $Object_{2F}$ are told what constraint to impose.

Figure 13.7 Refining the previously internal interaction



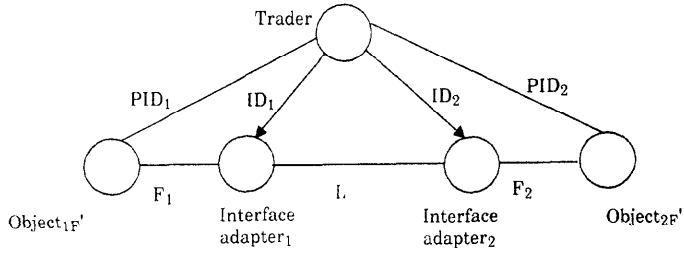
The next step in the transformation from canonical form of negotiation to the one of the trader is to combine the objects that perform the negotiation function into a single object called the trader. This is illustrated in Figure 13 .8.

Figure 13.8 Combining the negotiating objects into the trader



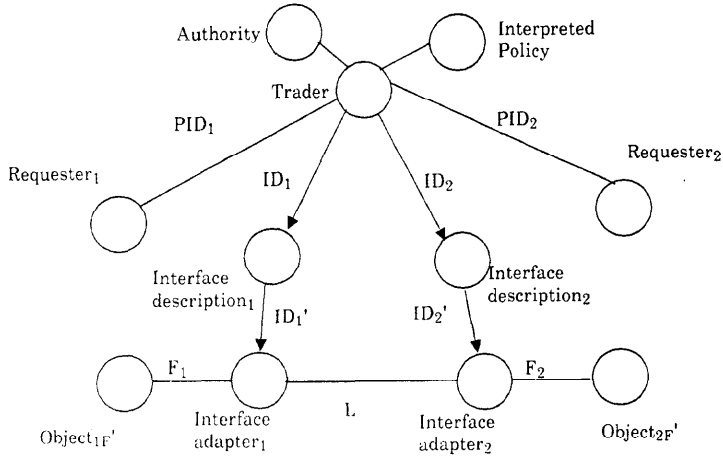
Both remaining objects take care of the rest of the interactions. They can each be decomposed into two parts. One part is only concerned with the application of pure function, the other is concerned with the adaption of the interaction alphabet to the connection environment. The objects $Object_{1F}$ and $Object_{2F}$ are thus decomposed to expose their interface adapters, as illustrated in Figure 13.9.

Figure 13.9 Exposing the interface adapters



It is then possible to make the interface descriptions held by the trader explicit in the diagram by showing them as objects. It is often possible to separate the requester part from the functional part of an object. The request or identification of the set of possible interface descriptions an objects can deal with are often prevalent at design time and are quite separate from the functional part of the object during run-time. If this assumption can be made, then the requester can be shown as an independent object. The trader may be further decorated with an interpreted policy and an authority, so that negotiation is subject to certain further constraints. The canonical form for the trader is shown in Figure 13.10.

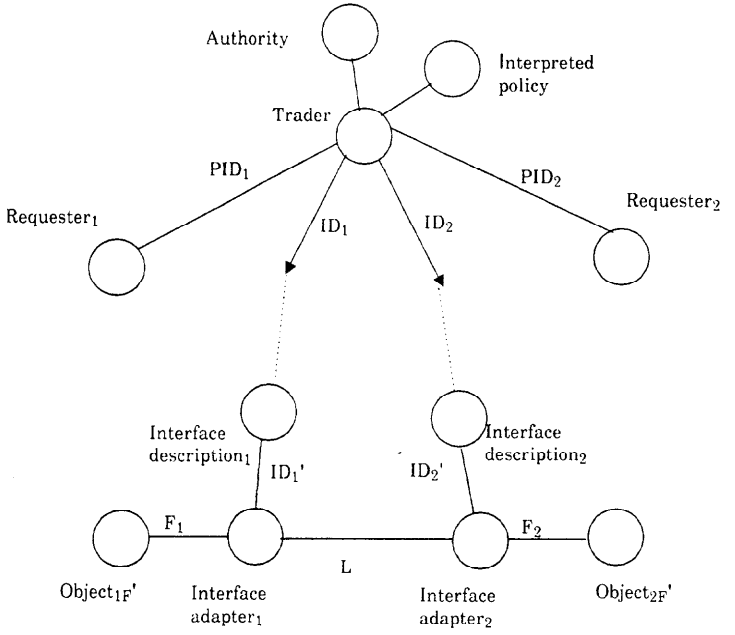
Figure 13.10 The canonical form for the trader, derived from Figure 13.10.



Finally, it is possible to analyze the system of Figure 13.10 in two separate parts. By application of "abstraction with epochs" (see Chapter 18), the diagram may be split as illustrated in Figure 13.11. The bottom half of the diagram can be identified as the canonical form for interface description and

interface adapter. It can be separately analyzed, but is subject to th constraints imposed on the trader with its associated objects. The trade with its associated objects can also be analyzed separately from any object that rely on it for their negotiations.

Figure 13.11 Abstraction with epochs applied to the canonical form o the trader



13.3.2 Trading and negotiation

Trading during design
Two objects that are being designed to interact with each other in a future system are designed by two different designers. During the design process both designers will need to meet and negotiate to reach or confirm an agreement over the conventions that must be applied for useful interaction between the objects. Usually the negotiations take place in stages of ever increasing detail, until finally the physical representation of the interactions are agreed upon.

Another example of this kind of trading is the standards making process of ISO, for instance. A standard is drawn up (agreed after negotiation) so that two open computer systems can communicate using the conventions (protocols) described by the standards.

Trading and resource allocation
In a previous chapter on resource management it has become clear that resource allocation involves the matching of available resources with requests from objects that wish to use them. An object that requires a print buffer, will not worry about which particular one it gets. The trading function determines which resource and therewith determines the interaction (time and place rather than type).

Trading and systems analysis
A design of a general invoicing package will allow a great many different kinds of invoices to be processed. Before a particular company or organization can use the package they will have to specialize the general invoicing package to suit their particular needs. The process of specialization determines the interactions between system and user and is a form of trading.

Versioning and archiving
A program library archive may contain many versions of programs that are all written to the same specification. When selecting a particular program, a policy is employed (e.g. last released version) to help the trader make a choice. The trader now determines the actual interaction between the rest of the system and the selected program from the library.

Trading at runtime and communications
Many of the protocols in the OSI 7 layer model involve trading. For the connection oriented protocols such negotiations are often part of the CONNECT-pdu exchange. A successful CONNECT-pdu sequence then implies a successful negotiation over communication conventions. The kinds of conventions that are negotiated are data representation by the presentation protocols, synchronization issues by the session protocols, buffer sizes and frequency of end-to-end acknowledgements by the transport protocols, routing protocols by the network protocols and packet sizes by the data link protocols. Sometimes negotiations can be reopened during the connection. Some transport protocols allow adjustments of buffer sizes and frequency of

acknowledgement to be tailored to the utilization of the transpoı connection.

When using connectionless communication protocols, the negotiatio procedure becomes more difficult to implement. Often negotiations will b conducted in a previous (design) epoch and there is no need to repeat them ε run time. Negotiation at runtime will require resources and possibly slo down data transfer. It does however increase flexibility over a situation i which the communication conventions are fully fixed.