Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

A Model of interception

Yigal Hoffner

Abstract

version 01: text as produced by YH

version 02: added figures etc from YH ascii files; no changes to existing text

APM.1142.00.02 **Draft** 18 February 1994

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

A Model of interception

A Model of interception

Yigal Hoffner

APM.1142.00.02

18 February 1994

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK	(0223) 323010
INTERNATIONAL	+44 223 323010
FAX	+44 223 359779
E-MAIL	apm@ansa.co.uk

Contents

1 Introduction

1.1 Objective

The aim of this document is to present a model of interception in the context of:

- authorities, administrations and management
- domains and boundaries
- the process of the co-operation setup
- the distributed development process.

This document complements RFA.002: " A Model of Object Co-Operation Set-Up".

1.2 Audience

Distributed system architects and designers who develop large scale heterogeneous systems.

1.3 Background

This document is based on earlier work presented TR.043: "ORB Inter-operability" [TR.043 93].

1.4 Introduction to interception

Whenever two objects co-operate there will generally be a gap between them. This gap can be characterised in terms of separation and difference:

- the gap between the entities will consist of physical SEPARATION
- the gap between the enetities will consist of a DIFFERENCE which is likely to manifest itself in a variety of ways:
  - — policy
  - — how things are carried out
  - — the information necessary for carrying things out and its representation
  - — lack of trust.

It is usually the case that a difference in authority, administration, information, mechanism or way of doing things, will inevitably be accompanied by physical separation of some kind (sub-routine, library procedure, remote procedure).

In distributed systems there is a need for support for:

- co-operation between clients and servers: such co-operation may require inteaction to cross a boundary. The boundary marks adifference between two objects, which has to be bridged
- prohibiting certain interactions: Prevention may be necessary where the crossing of boundaries may cause loss of integrity due to failure, breach of security or may be unwanted for commercial reasons
- observing events of interest: Monitoring is often required where the recording of interactions is necessary for traceability, accountability, debugging, dependability, and garbage collection for example.

In all of these situations, there may be a need to modify objects or insert adaptors between them in order to make them compatible with each other, or to prevent certain undesirable interactions between them from taking place, or to monitor interactions. This general process is referred to as interception.

Interception is therefore about the information, mechanisms and processes necessary for:

- detecting attempts by entities to establish co-operation and actually co-operate
- determining the differences between the entities and whether any intervention is ncessary
- inserting the necessary mechanism. The mechanisms chosen will depend on whether support is required for:
  - — enabling the crossing of the integration boundaries
  - — disabling the crossing of boundaries
  - — monitoring the crossing of boundaries.

In addition the mechanisms will depend the kind of boundary being crossed.

- acting on attempts to interact across the boundary.

1.5 The relationship between interception and distributed system issues

Note: Possibly shorten this and move most text to the relevant sections in last chapter!

Interception and object orientation are closely related. The encapsulation which objects require is based on creating fences around data and procedures, whilst at the same time allowing their use from outside the encapsultion boundaries, through well defined and protected gates. In order to ensure proper interaction and also secure the object's integrity, any attempts to cross its encapsultion boundaries have to be detected and dealt with.

Separation into modulescan be used as a technique for dealing with complexity. Similarly, differences often manifest themselves in a physical separation between entities. Thus bridging differences usually involves overcoming physical separation, and hence the reason for the close link between interception and communications. In the context of the software development proess, the physical separation usually manifests and resolves itself by one of the techniques used to achieve modularity: macro expansion, procedures, library routines, remote invocation (remote procedure calls). These in turn use one communication medium or another (JSR, IPC, RPC). Hence the close relationship between interception, the software development process and communication.

Interception may take place in different stages and at different places of the development process: design, implementation and set-up time, as well as run-time where interceptors may have to be created, inserted and dismantled dynamically.

Interception requires information about the entities which have to co-operate. Such informationcan be derivedfrom the trading process [?]. Trading facilitates the exchange of information between the client and server and their respective environmentsnecessary for co-operation. Thus, interception and trading are closely related.

There is another link between interception and trading and binding. Trading and binding are part of the process of creating new links between objects. It is precisly where such links are being created that interception is necessary.

Binding is the process of linking together separate entities in the system using the resources available [TR.043 93]. Where interception will be needed it will be necessary to create and insert the appropriate mechanisms. Inteeception and binding will therefore be closely linked to each other,

Creating the components of interceptors will involve the use of templates and factories and is therefore also related to configuration management []. In addition, where interception can be used to monitor activities such as the passing of Ifref's, thereby providing information necessary for garbage collection [].

When objects migrate they will cross one, or more likely, several types of boundaries. The crossing of these boundaries will have to be detected and dealt with in order to facilitate migration.

1.6 Basic terminology

Note: This material may be more appropriate in the paper on Boundaries and Domains

In order to investigate the topic of interception, it is necessary to define the necessary terminology. The above description of interception can then be re-examined in the light of these terms.

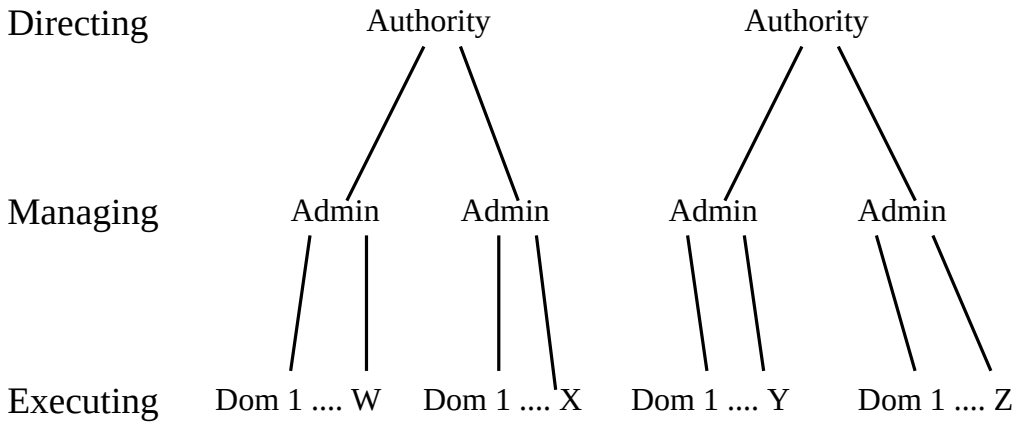
For the purpose of this work an *authority* is a directing body capable of decision making and which has the power to confer rights concerning creation, ownership and control of a set of objects.

An *administration* is a bureaucratic term used to alternatively denote a set of people who are being administered, the set of procedures used to administer, and the subset of people who manage the process. Policies determined by an authority are usually translated into a set of procedures which 'administer' the policies.

Authorities are likely to be in charge of a number of different administrations. The relationship between: "*authority* - *administration* - *domain*" is parallel/ synonymous to the relationship between: "*directing* (decision making) - *managing* - *executing* policy (policy enforcement)" (Figure 1).

A *domain* is a set of entities which are under a common administration and authority [SLOMAN 89]. This implies that a common policy applies to the entities resulting in a pocket of homogeneity with respect to the issues determined by the common policy. The essential characteristic of a domain is the relationship between objects and authority. The area which is defined by

Figure 1.1: The relationship between authorities, administrations and domains



the extent of the homogeneity which exists by purpose, ie., by the influence of an authority and its application of its policy by the administration, is called a domain. There may be areas of homogeneity which are not co-terminus with domains of authority.

A *boundary* delineates a domain, ie. the line whichdefines the extent of the authorities' control and beyond which other systems exist which may or may not adhere to the same models, policies, mechanisms and general way of doing things.

There are different types of domains and boundaries depending on the aspects which may differ between systems [APM 1039 94].

Systems need to incorporate facilities for crossing these boundaries where this is feasible and permissible and also prevent it where it may lead to loss of integrity.

A *fence* is a boundary in a physical sense. Fences are set in order to restrict or control entry and exit into a domain. The setting up of fences is usually closely linked to issues of trust.

A complementary notion to that of a fence is a *gate* which is a recognized and controlled opening in a fence. In computers, fences and gates are almost invariably associated with communications mechanisms due to the fact that the boundaries between entities are usually accompanied by physical separation, ie. they manifest themselves by a subroutine, a macro call, a library procedure or a remote object.

A reason for distinguishing boundaries from fences is that a boundary may have no manifestation, or it may have one or more fences associated with it.

The relationship between: a boundary and the way it may be implemented as a fence is analogous to the logical separation when modelling objects, and the manner in which the separation is encapsulated.

The same boundary may have different physical manifestations as a fencedepending among other things on how objects in each domain are designed and implemented. Thus a boundary may have different manifestations/implementations in space and time.

Note: No man's land concept is not quite worked out!

No man's land is the area outside a boundary which is not necessarily within another domain. Thus the crossing of a boundary takes the activity to no man's land in the general case. No man's land may be regarded as a domain where standardization may be applicable and useful. No man's land is usually where the communications lie.

1.7 Example to explain terminology

An analogy may explain the concepts behind the terminology.

A domain is analogous to a region or a country. A boundary is analogous to a border between two countries or regions. A fence may be a barbed wire construction which marks a border. A border between two countries, may be unmarked by a fence depending on the agreement and the differences between the two countries. Alternatively, there may be a single fence between the countries, or even two sets of fences, one for each country. The relationship between the countries may even necessitate an area where neither are in control - a no man's land. This has to do with issues of trust.

There are situations where there will be no fence yet a gate will be present. Such a gate may be anywhere and not necessarily related to a geographical notion of a boundary. An example is a currency exchange post to be found in a city, nowhere near the border between two countries.

Different types of boundaries exist between the countries such as judiciary, monetary, employment, etc. Some of these may be unmarked by a fence.

An interceptor is analogous to a gate and a customs guard, which identifies the items passing through the gate.

Examine (note interception as well as migration issues are described):

- currency exchange
- declaring goods at customs
- use of Id cards and passports
- and applying the law to a foreigner (parking tickets)

Note: Examine the fitness of the terminology in the context of:

- software development
- public telephone systems
- wireless phones
- security, performance, dependability, monitoring, debugging

Note: Questions which arise from the example:

- what difference does it make to a client and server what the agreement between the domain managements and authorities is?
  - intermmediate languages
- does it make any difference whether it's a federation or hierarchy?

1.8 Interception and distributed system assumptions

Three assumptions prevent entities in one domain from co-operating with entities in another domain:

- single domain assumption: the assumption that my domain is the only one in existence
- the closed world assumption: recognition of the existence of other domains, but without any intention of relating to them.
- sameness assumption: that everyone has the same models, procedures and mechanisms as me.

These assumption cause two major problems as far as interception is concerned:

- where there is no recognition of the existence of other domains and the possible differences between them, it is difficult if not impossible to define boundaries and consequently to erect fences at the points of contact with other domains
- even if boundaries could be identified and and fences erected, it is possible to have a situation where the influence of the assumptions is such that the design of applications will prevent co-operation with external objects.

Note: Does someone have an example from the world of computers? The Golan heights is an example from politics!

1.9 Interception pre-requisites

Pre-requisites for succesful interception are:

- not to have restricted the design by making any of the assumptions discussed in §1.8. Interception is possible only in a system which recognizes the existence of other domains and is willing to co-operate with entities in other domains. This is the designer's responsibility.
- reach agreement on co-operation between the authorities. This is the responsibility of the authorities and their administrations
- being able to deal with the differences between the systems separately, at least in the conceptural level. The implementation of interception may be optimized to deal with several issues in an aggregated form (see stub generators as an example of an interceptor that deals with more than one issue at a time).

2 The process model

2.1 Major stages in the interception process

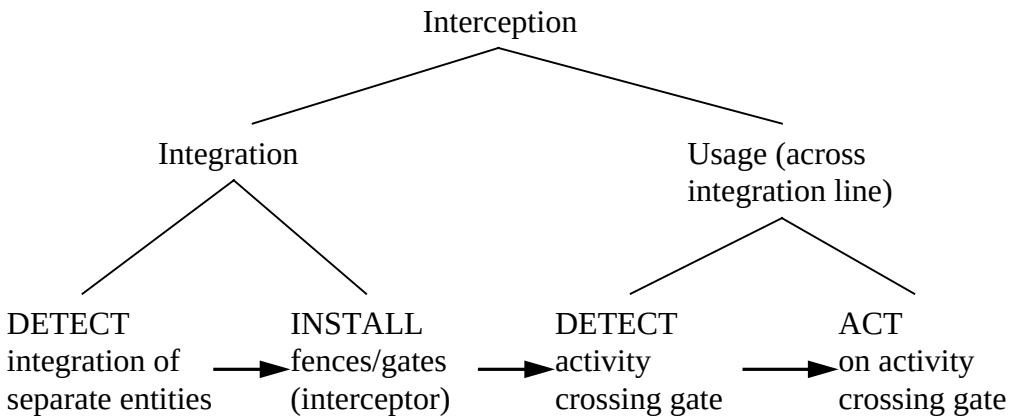
Note: Note that the story is recursive and applies to boundaries and also to fences

Note: If the difference between creation/insertion and active inteception was called differently it would be possible to describe the recursion more accurately.

There are several major stages involved with interception (Figure 2.1):

- dealing with theco-operation between entities:
  - — detection of an attemept to integrate
  - — installing the necessary fences/gates
- dealing with the activity crossing the boundary:
  - — detection of an activity attemepting to cross a boundary
  - — acting on the activity itself.

Figure 2.1: The major stages of interception



2.2 The interception process in detail

Provided the pre-requisites discussed earlier (section ?) are fulfilled it is possible to implement interception.

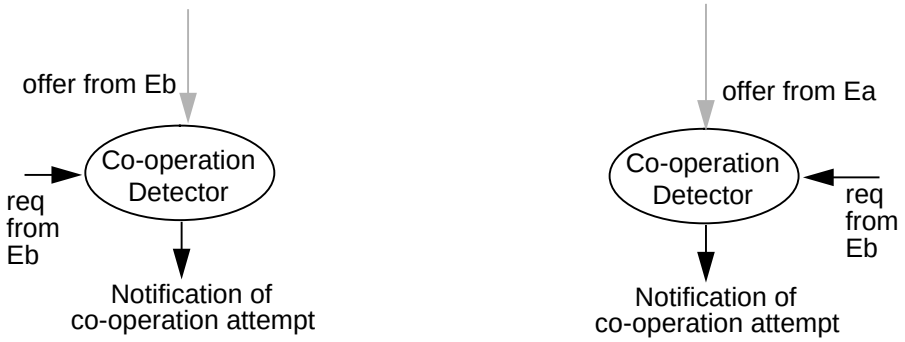
An entity in this section may be an actual object or a domain.

In order to achieve the desired co-operation, the authorities have to:

1. Integration:

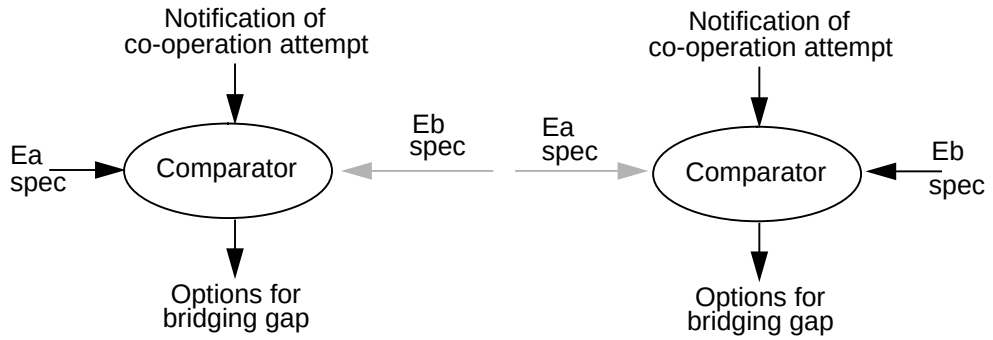
- detect integration attempt and recognize the existence of other domains (Figure 2.2)

Figure 2.2: Detecting attempts to establish co-operation between two entities



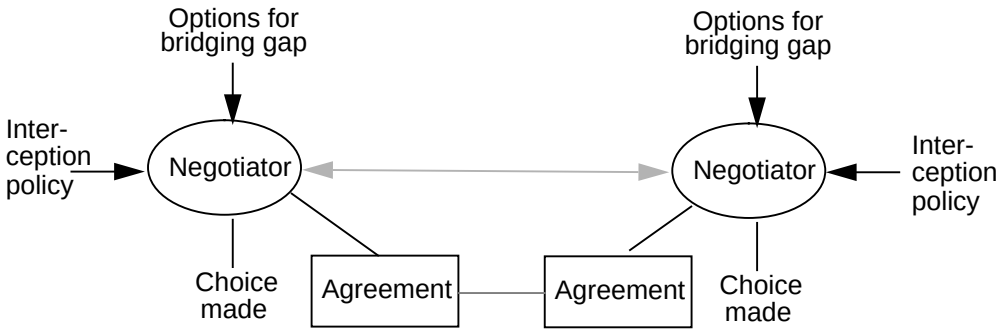
- recognize the boundary and fences between the domains
- get the specifications of the objects in their respective domains (trading issue)
- determine the differences between the two objects and domains and the options for bridging the gap (Figure 2.3)

Figure 2.3: Getting the specifications of both entities and comparing them



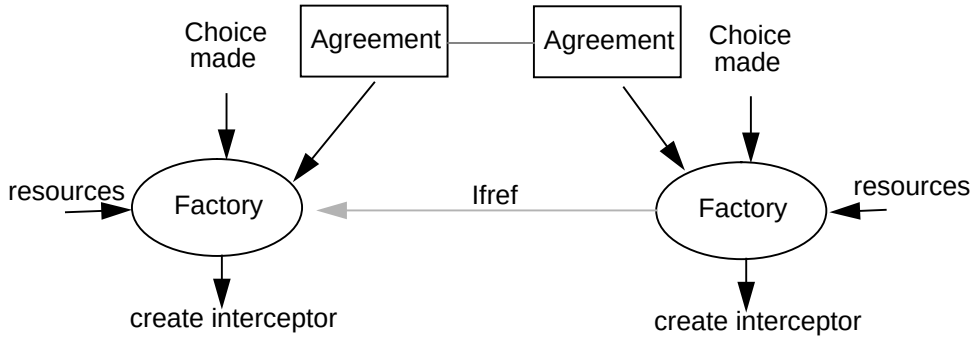
- negotiate with the parallel interception process taking place in the other domain and reach agreement on what has to be done to bridge the difference and how (Figure 2.4)

Figure 2.4: The negotiation process and the resulting agreement



- use the agreement and information from the comparison to construct the interceptors to bridge the difference (factory issues) (Figure 2.5)

Figure 2.5: Creating the interceptors



- insert the interceptors between the two objects at a gate in the fence (binding issue) (Figure 2.6): by giving Is the Ifref of the server and giving the client the Ifref of Ic, and passing the Is Ifref to Ic through the Diff-detector or the Interceptor factory

Figure 2.6: The resulting configuration of client, server and interceptors

The resulting model is shown in Figure 2.7.

2. Usage across the integration boundary (Figure 6):

- detect the crossing of a gate o act on the activity crossing the gate according to the type of boundary being crossed. The resulting mechanisms may act as:
  - — enabling technology: performing the appropriate transformation for overcoming the differences between domains
  - — disabling technology: refusing entry and/or exit from domains where this is not permissible
  - — monitoring technology: observing, recording and notifying attempts to cross boundries.

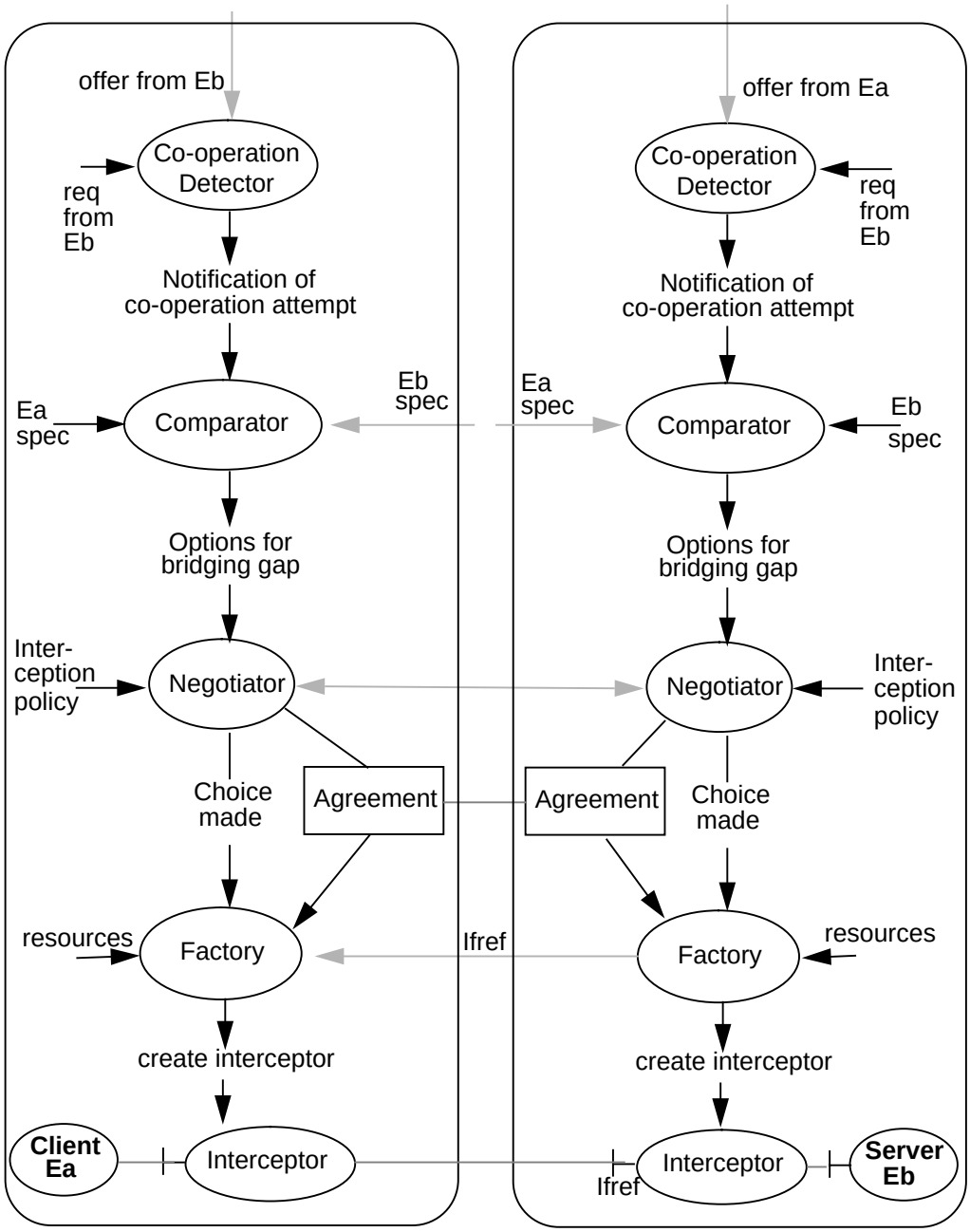
2.3 Symmetry of the model

Both sides have to go through the same steps, each with the respective information and make decisions and take actions according to their:

- policy concerning co-operation and interception
- resources available

The resulting model is symmetric from the point of view of the two entities (Figure 2.7).

Figure 2.7: The general (symmetric) model of the process of interception



The symmetry in the model ensures that each domain has control over it's own side and cannot be forced by the other side to do anything.

Note: Leave following para's for the time being!

Note: Figure 2.3: Getting the specifications of both entities and comparing them. The result of the comparison is fed as input to the negotiation process

Note: The options for bridging gap will depend on: o policy concerning co-operation and interception, and security o resources available (link to configuration management

Note: Figure 2.4: The negotiation process tries to establish what the options which are acceptable to both sides are

Note: Figure 2.5: The agreement from the negotiation process can be used either to: o input to the factories for producing the interceptors o point of arbitration between the two entities (eg. security, renumeration)

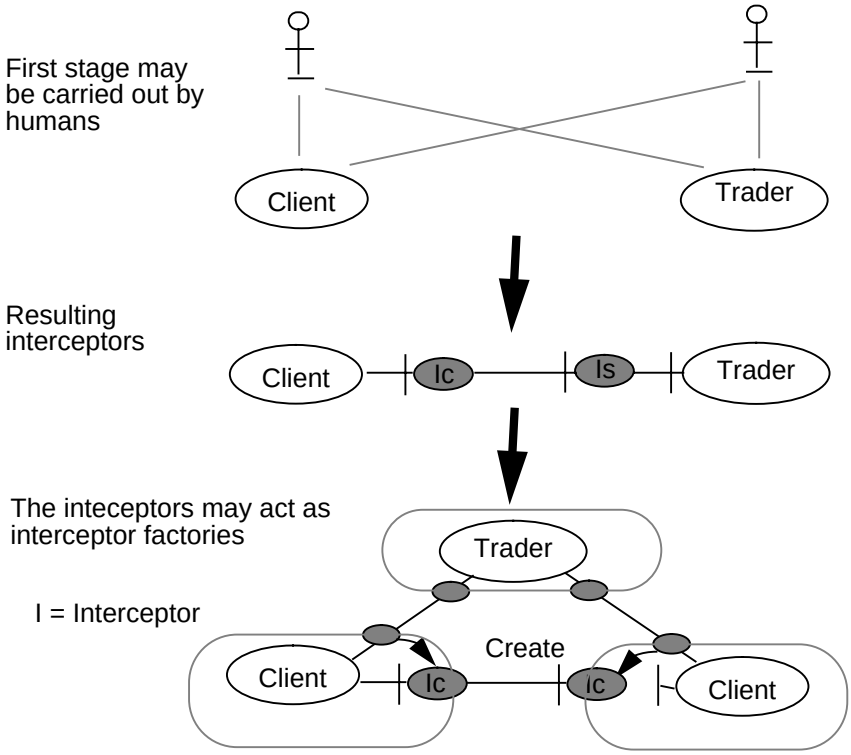
Note: Figure 2.6: The resulting configuration of client, server and their respective interceptors

2.4 The recursive application of the model of interception

Interception can be applied between two objects in such a way that the resulting interceptor inserted between them acts as an Intercepror-factory.

The story of interception can therefore be recursive, as shown in Figure 2.8

Figure 2.8: Recursive application of interception



Note: Put figures to show this!

Note: If the difference between creation/insertion and active inteception was called differently it would be possible to describe the recursion more accurately.

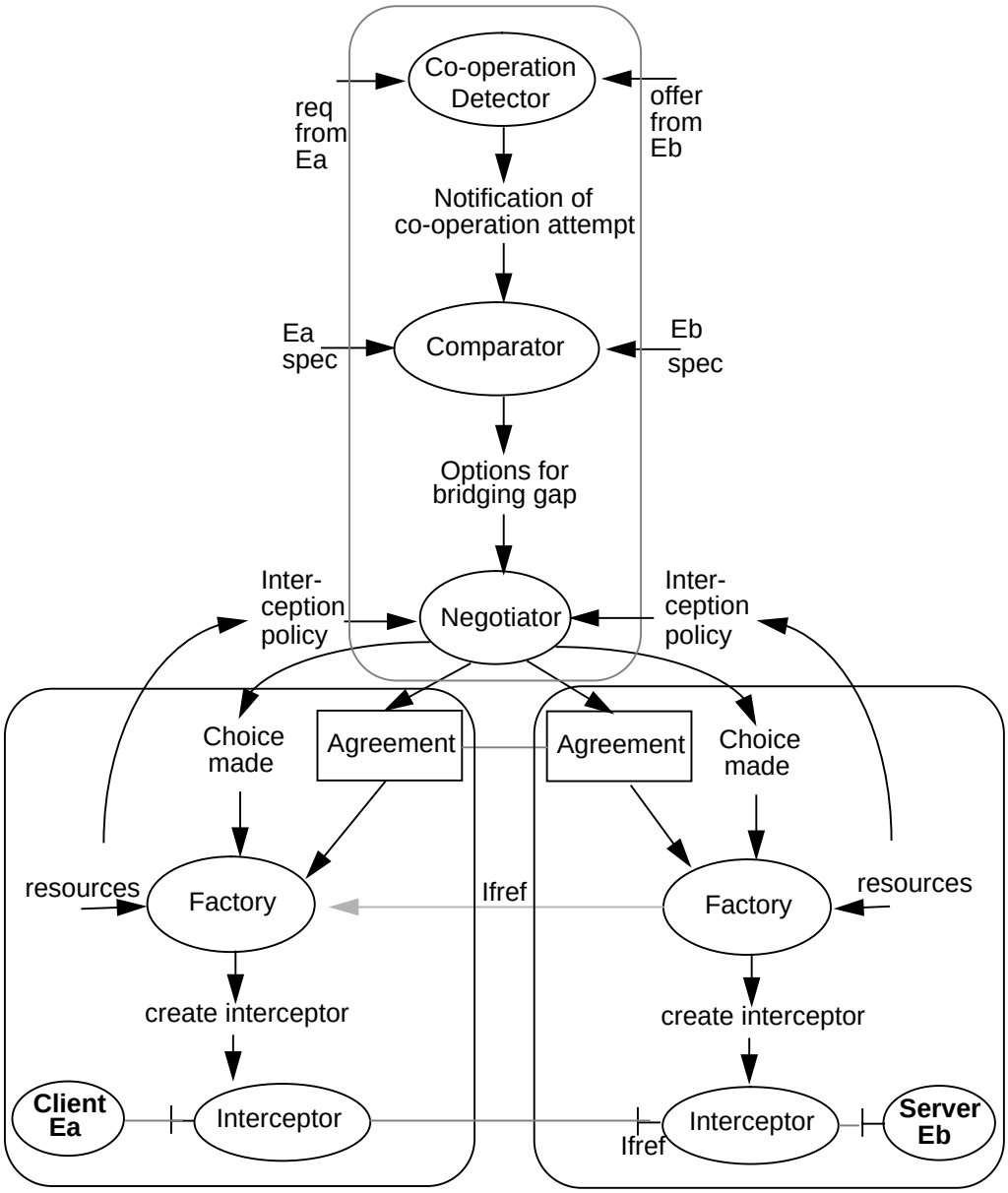
2.5 Special cases of interception

Simplifications of the model are possible in different circumstances. As the processes can be carried out at different epochs, one way of simplifying the run-time stage is to carry out as much of the process prior to run-time. This simplifies the run-time activity but reduces choice (early versus late binding problem).

Examples of specializations:

- where a single authority/administration is in charge of both entities or two authorities in close co-operation (Figure)
- agreement to use standards
- prior agreement between the authorities
- prior choice which renders one or more of the processes unnecessary at run-time (Figure)
- where sufficient trust exists between two domains to allow a single interceptor to deal with things on behalf of both domains (Figure)

Figure 2.9: The trusted interceptor



Note: See TR.038 Information model.

3 Trading, binding and interception

The processes of trading and binding are concerned with the setting up of new links between objects. Interception is about enabling, disabling or monitoring interaction through such links. It follows that these topics are intricately related.

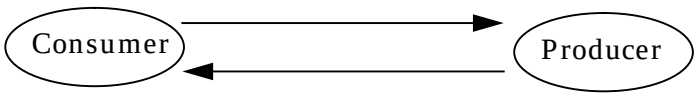
Note: Interception and dynamic creation of services are closely linked to each other. In the case of interception, the end result of successful inteception process is to produce an adaptor and insert it. The end result of trading if the server or client do not exist is their creation and insertion.

3.1 Interception and trading

This section is about the link between the interception model and the process of trading.

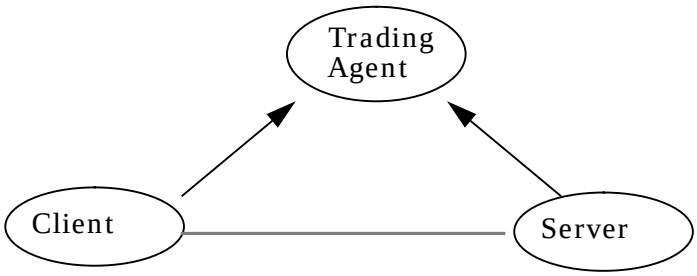
Trading is concerned with passing information between clients and the servers concerning co-operation (Figure 3.1).

Figure 3.1: The general (symmetric) model of trading



This is often implemented in the form of a trader with a client and server performing export and import operations on [?] (Figure 2). Thus, a starting

Figure 3.2: The general (symmetric) model of trading



point to the process of interception is configuration containing the client, server and their respective domains, and the trading agent. Since the interception process can be applied at different epochs, we assume that the differences between the client and trading agent, and the server and trading agent have been sorted out at a prior epoch. This results in a simplified model (Figure ??).

3.2 Interception, trading and inteceptor factories

Editorial: combining interception with the trading process provides the way to ensure successful co-operation between objects.

3.2.1 Optimizations (combining trading with stages of the interception process)

These are design decisions as far as the basic model is concerned.

The trading process provides another point where the differences between the client and server can be identified. The trading process can then make a decision or conduct a dialogue with the interceptor factories to create the required interceptors. This results in a more centralized model of interception (Figure ??).

3.3 Interception and binding

3.4 Implementation issues and examples

There are many ways in which this general model can be implemented depending on resources and constraints and which areas require and can be optimized.

In ANSAware [ARM 93], for example, binders are responsible for the interception concerning communications. Thus, they have to choose from a range of avauilable communication protocols. Rather than having a dialogue, the interface reference passed from the server to the client (usually via the trader) specifies the options which the server is offering. These are set up by the server binder so that regardless of the one chosen by the client binder, the server side can deal with the incming message.

Another example of interception concerns the difference in programming languages and data representation between systems. Wherever invocations on remote objects are performed, stub-compilers can intercept the invocation code and create and insert the necessary stubs to be used when a remote invocation takes place [ARM 93].

Note: Any more examples?

- the object management community has experience in translating CMIS/ CMIP specifications into Corba IDL (and vice versa)
- our previous experience in the Harness project showed one way to generate application level gateways between applications in different distributed systems environments (ANSAware and DCE)
- the use of IDL and stub-compilers to create stubs has been used in several places such as ? [?] and ANSAware [ARM 93]. Stubs and the RPC protocol bridge three types of gaps:
  - physical distance
  - differences in machines
  - differences in programming languages and data structures and representation.

4 Types of interceptors

Interceptors will vary according to:

- the type of boundary they are associated with
- the action taken by them.

4.1 Interception and boundaries

The following is a list of kinds of boundary [RFA.001 94], and the interception scheme used to bridge them:

- authority: contracts,lawful procedures, arbiters -
- administration: - policy convertors - management protocol conversion: CMIS/CMIP, SNMP and Corba IDL o renumeration: - exchange posts - banks
- infrastructure:
  - comms protocols: Binders
  - transparencies: Binders
- application client-server:
  - application semantics: Translators?
  - QoS:
    * security: Security gates, authentication servers
    * performance:
    * depenability:
  - programming language and data representation: stub translation and on the wire format (OTWF) [APM.1042 93]
- models and naming [AR.01 93]: - file names: name servers - name translators

It should be possible to deal with the different areas separately although optimizations may motivate their aggregation.

4.2 Interception and action

Common to all interceptors is the act of detection of a boundary crossing. However, upon detection of a crossing, interceptors will take different actions. The following types of interceptors are therefore distinguished:

- Interceptor-Monitor: observers, records and notifies interested parties such as management, for example, of the crossing of a boundary. Important aspects of interactions which may be useful to monitor and record are ones concerned with, for example: - passing interface references for garbage collection purposes - transactions concerned with money

- Interceptor-Bouncer: rejects the attempt of an interaction to cross a boundary either:
  - — from outside in
  - — from inside out
- Interceptor-Adaptor: carries out the necessary operations such as conversions, re-direction or checks (authentication in the case of security boundary) when a boundary is crossed
- Interceptor-Factory: detects an Ifref crossing of a boundary and is capable of generating a proxy Interceptor/Adaptor
- Interceptor-Redirector: re-routes an activity crossing a boundary.
- Interceptor-Checker: checks for the rights to cross a boundary (authentication in the case of security boundary)

5 Interception and other issues

5.1 Interception, communication and the development process

The separation of components from each other occurs for two reasons: o separation of issues/things to deal with complexity o common issues/things are often encapsulated together.

Physical separation often marks the dealing with different issues or the dealing with the same issues differently.

Different techniques which support separation are used in the software development process (Figure 5.1) :

- macro call: “in-line” expansion by a pre-processor
- subroutine call: compiler implements a call in the same address space (Assembly language JumpSubRoutine - JSR)
- library procedure: linker implements a call to another address space on the same node using Inter Process Communication (IPC)
- remote object: trader/binder facilitate a remote procedure call (RPC) to a node which is physically remote.

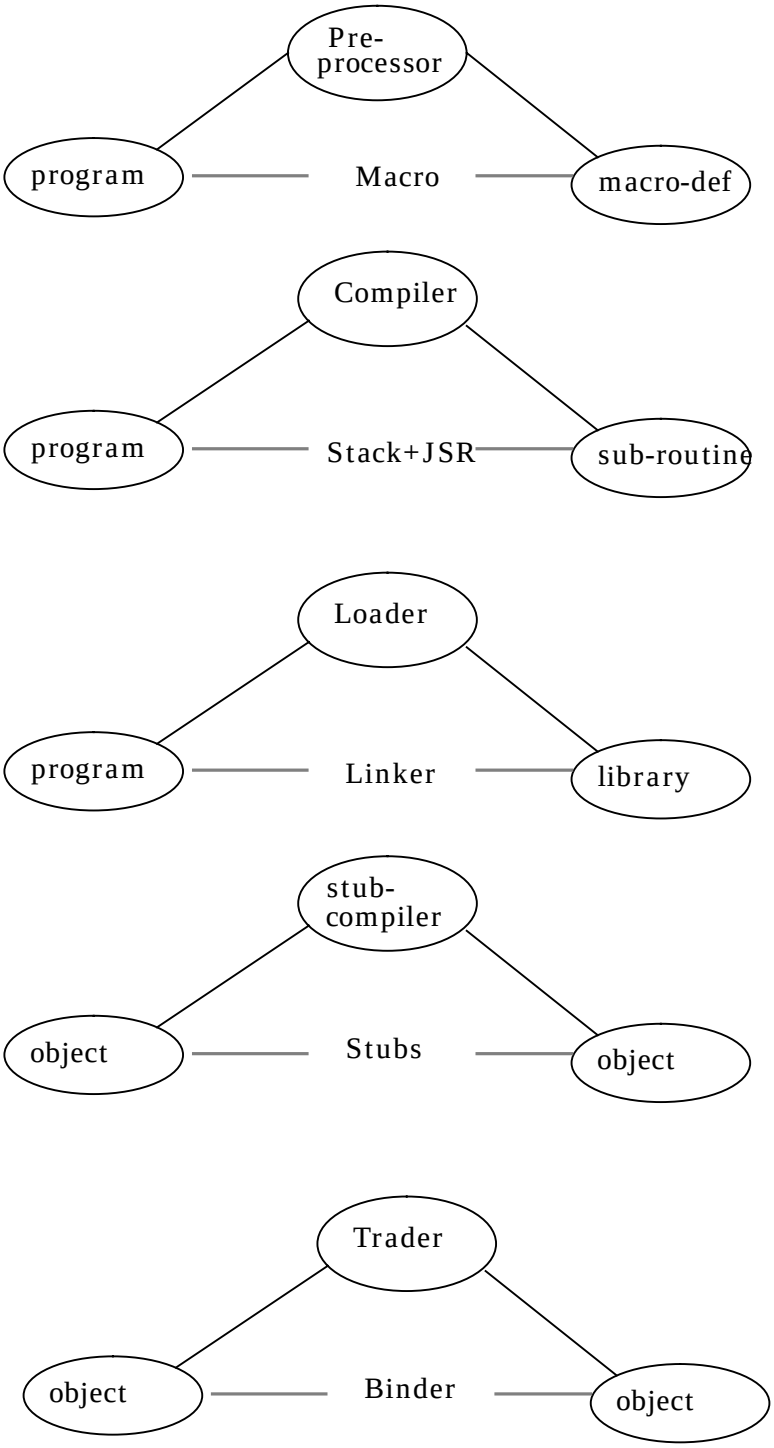
5.2 Interception and configuration management

5.3 Interception, interface references and garbage collection

One of the important functions registers the passing of ifref:

- Interceptor/Factory
- Interceptor/Adaptors created by Interceptor/Factory
- configuration management
- comms objects
- objects holding the Ifref
- objects handing out the Ifref: servers, traders, binders.

Figure 5.1: Different types of integration techniques and tools



References

[ANSA 91]

ANSA: A Systems Designer's Introduction to the Architecture, APM Ltd., Cambridge U.K., April 1991.