**Poseidon House**
**Castle Park**
**Cambridge CB3 0RD**
**United Kingdom**

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

**ANSA Phase III**

# Matchmaking Service: information model and repository (TC presentation)

**Rob van der Linden**

**Abstract**

Rapid deployment and flexible configuration of new and existing services is a requirement in a market place characterised by shortening market windows and highly personalised services. Software services are increasingly at the hub of modern systems.

A Service Management Architecture is proposed to support increases in productivity required by service providers in the telecommunications service business and elsewhere.

An important element of the service management architecture is the availability of service specifications and the accessibility of service instances. The MatchMaking service provides access to service specifications.

This presentation describes the work done under work item F3 in the ANSA Phase 3 project: the analysis of information which may make up a comprehensive service description and the prototype MatchMaking Service based on Iona's ORBIX and HP's Allbase.

APM.1293.00.01 **Draft** 6th September 1994

Briefing Note

**Distribution:**
**Supersedes**:
**Superseded by**:


# Work package F3

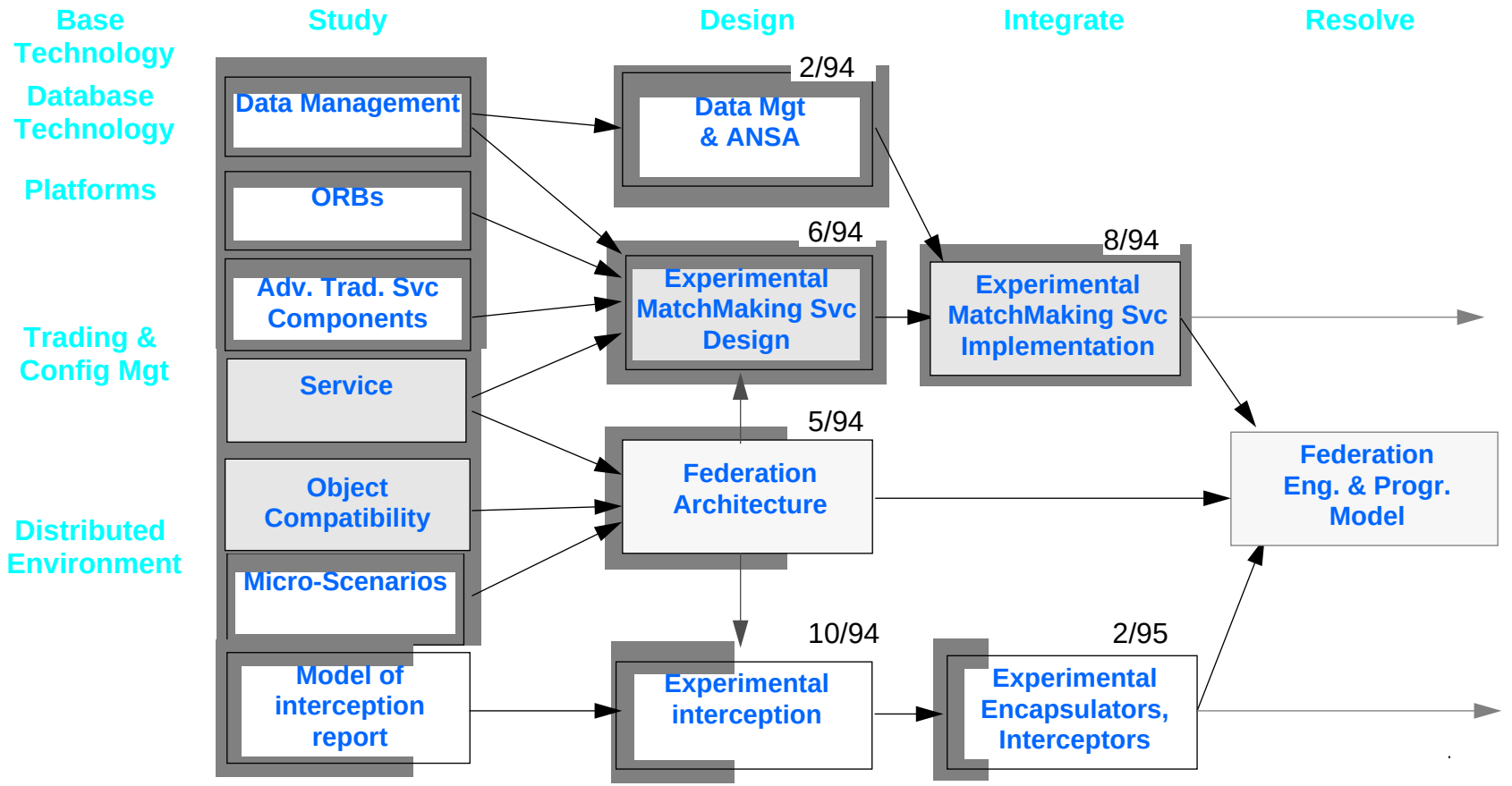
# MatchMaking Service

**Rob van der Linden & Gomer Thomas**

**(reporting on work done by
Mike Beasley, E. Jane Cameron, Yigal Hoffner,
Rob van der Linden and Gomer Thomas)**

- **Information Model**
- **MatchMaking Service design and implementation**

# Federation Plan



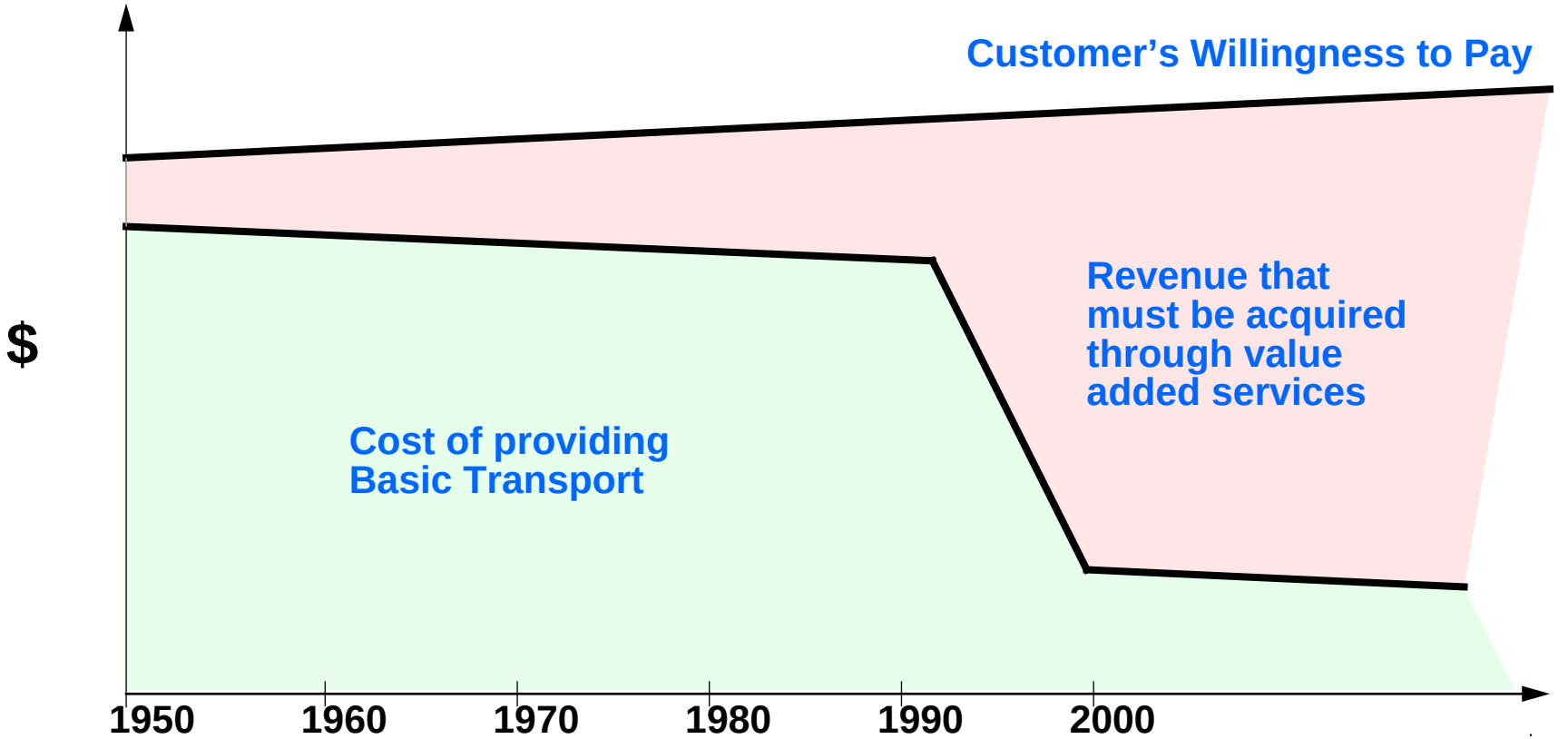

# Objective of this talk

- *review the work done under workpackage F3 & remind you of its background*
- *get your input as to how to package the result*
- *use the resulting package as a basis for technology transfer*
- *point the way forward*

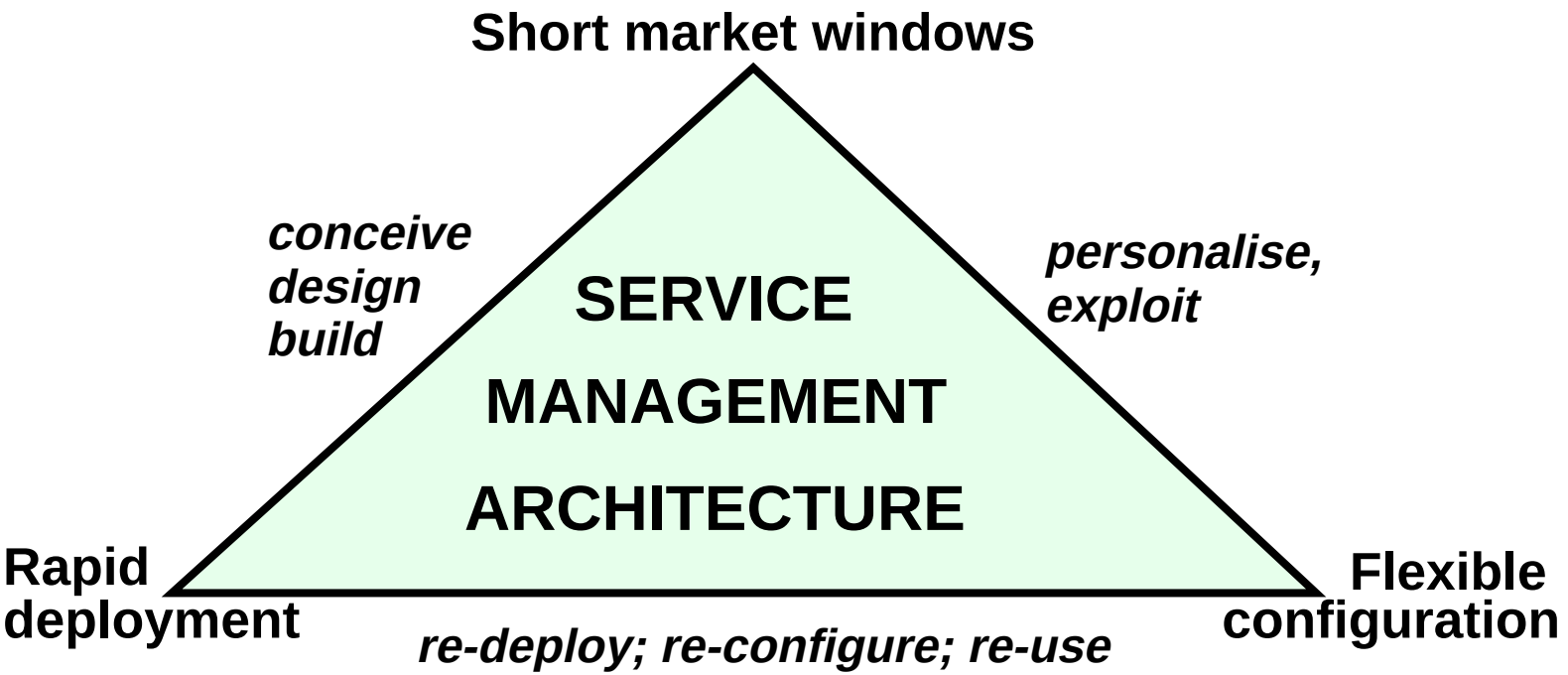
- **Background**
- **Information Model**
- **MatchMaking Service**
- **The way forward**


# Background

# The business challenge



# A service centred view




# Open Software Architecture

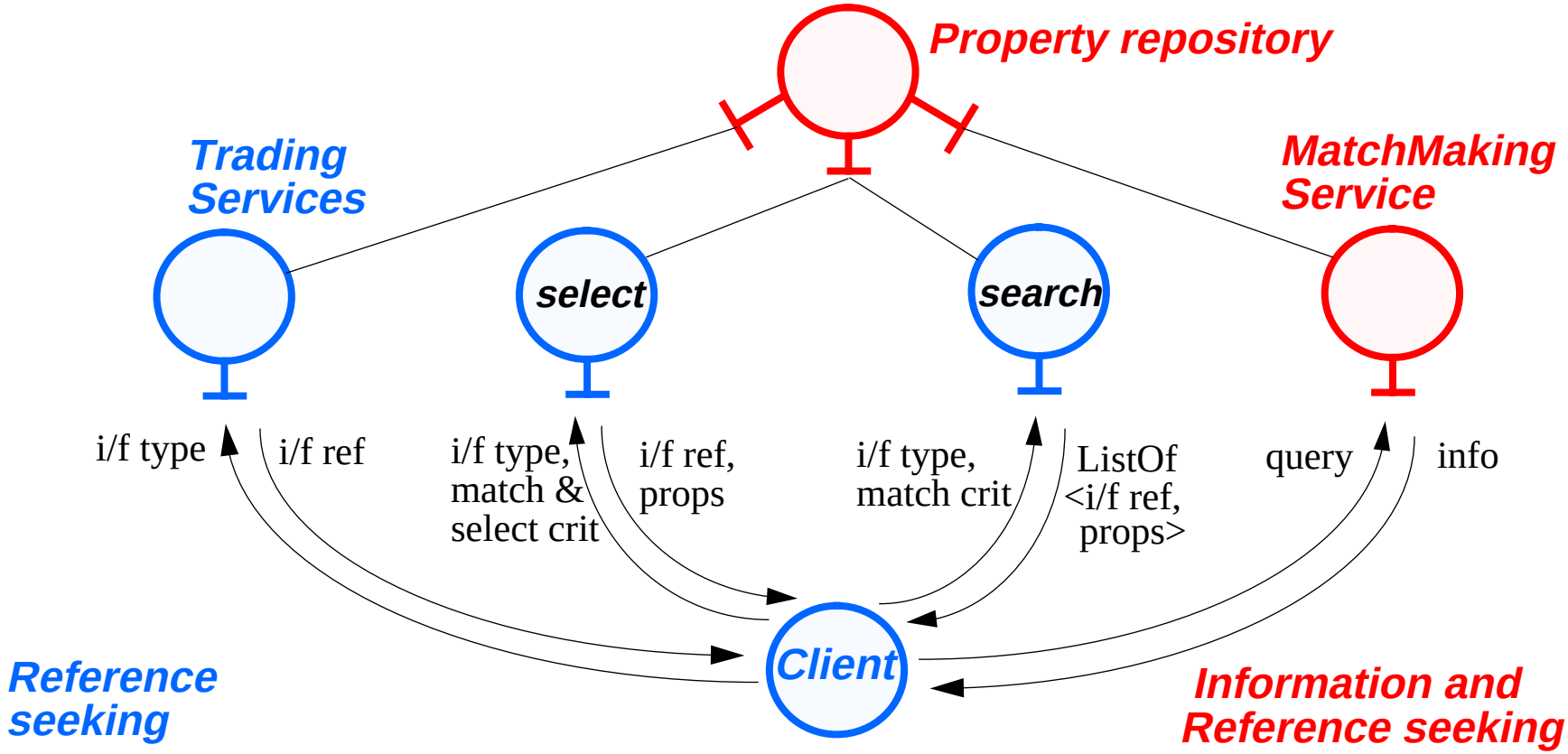
- ***Systems carry their own model so they can evolve***
  - **specifications of components are visible from the system**
  - **implementations of components are available in the system**

## but

- **what information should be held**
- **how should it be structured**
- **how should it be managed**
- **and how will it be used**

**?**

# Trading and MatchMaking



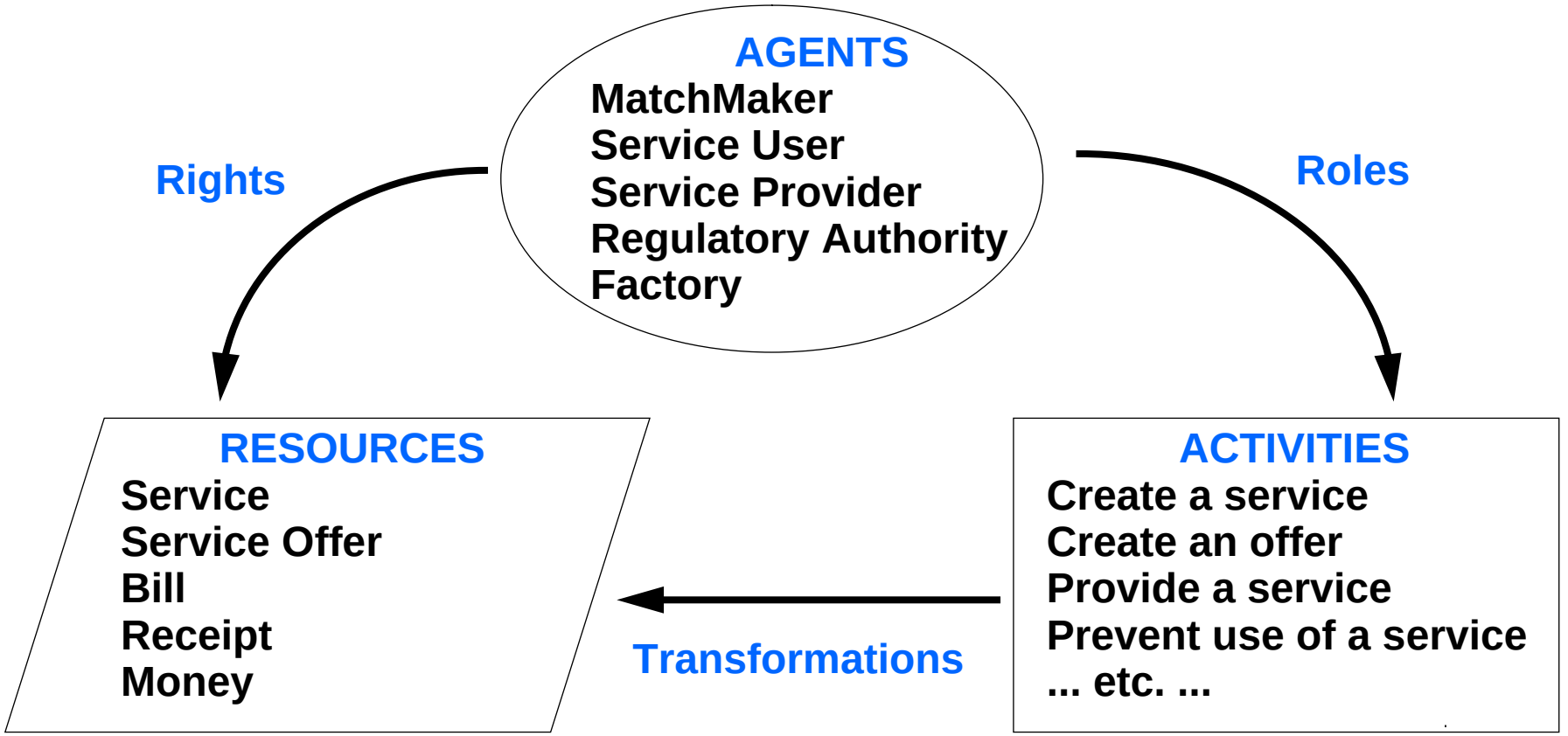

# Information model

- What information
- How to represent it

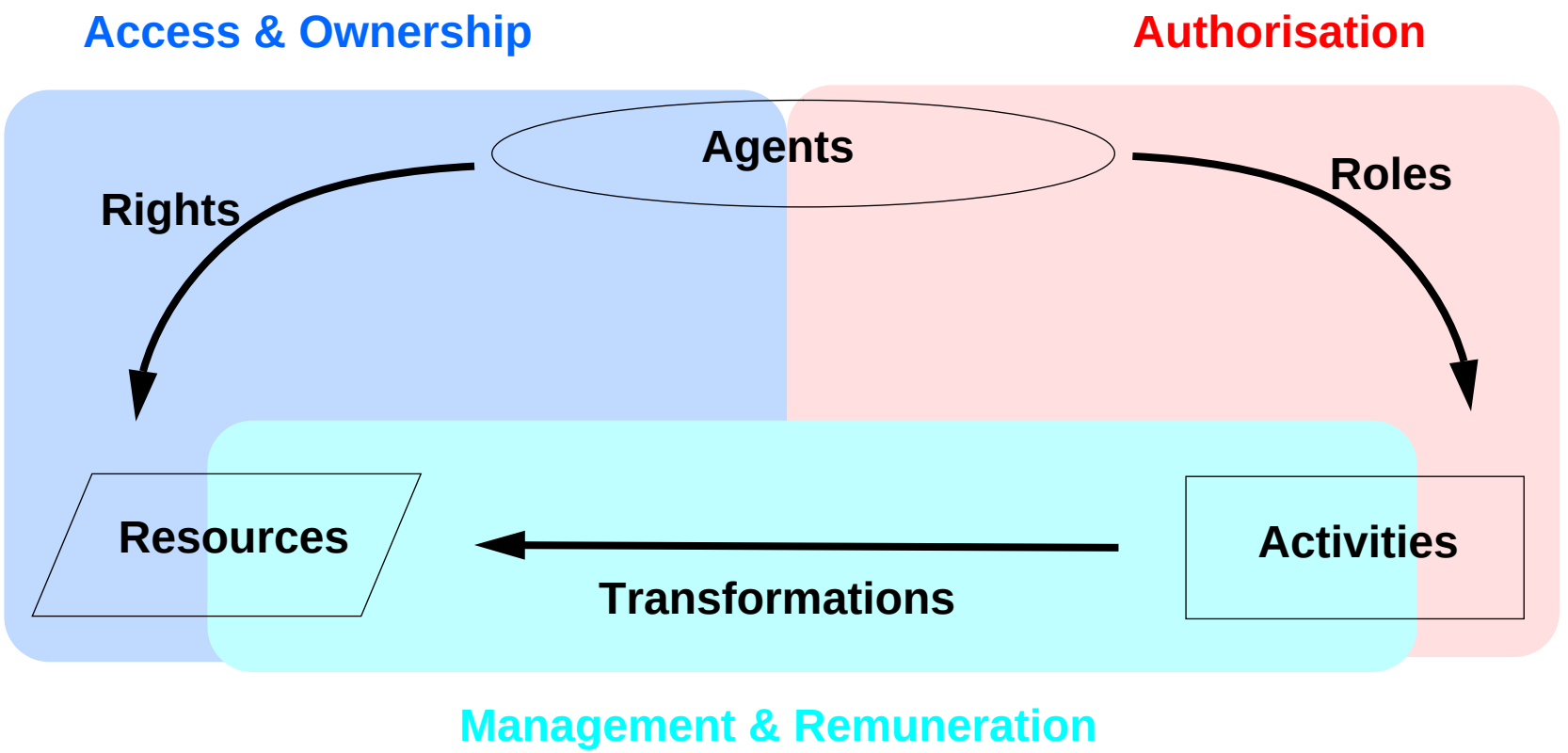

# What information?

- *Specific information to do with boundaries (differences)*
- *Federation manifesto identified key aspects of interoperation across boundaries:*
  - **Authority (institutions)**
  - **Accounting & billing (remuneration)**
  - **Management (administration)**
  - **Infrastructure (technical)**
  - **Application (interface descriptions)**
  - **Model (names and meanings)**
- *but who are the agents, what resources are involved, what activities take place?*


# An enterprise model for MatchMaking



# Enterprise model (2)




# Remaining boundaries

- ***Model:***
  - **Naming and meaning (information viewpoint)**
- ***Application:***
  - **operational and stream interface descriptions: (computational viewpoint)**
- ***Infrastructure:***
  - **QoS and Management of QoS (engineering viewpoint)**
  - **binding and streams (technology viewpoint)**
- ***An example: What real problems do we know about?***


# Representation of information

- ***Basic and constructed data types for MatchMaking Service were derived***
  - **we were satisfied that the types of information identified could be encoded**
- ***The video on demand example is a first step to test the information model***
  - **it was based on the key aspects of interoperability identified in the manifesto**
  - **this scenario should be completed**
- ***Other scenarios will be introduced***
  - **e.g. electronic business**
  - **interception requirements**
  - **real time scenario**


# MatchMaking Service


# MatchMaker Prototype Objectives

- ***In general: support ODP Trader functionality***
- ***But:***
  - **support all properties needed for federated interoperability (all projections, all epochs --> complex, extensible type system for properties)**
  - **support instantiable, parameterized services/interceptors**
  - **implement with currently available commercial infrastructure**
  - **provide scalability to very large collections of services**


# How About ODP “Export” Operation?

- ***Input parameters:***
  - Service interface type
  - List of property <name, value> pairs
  - Service interface reference
  - Policy Controller (Monitor) interface reference (optional)
- ***Challenges:***
  - What is the type of this operation (given multiple types for properties)?
  - How does MatchMaker know the type of each property?
  - How does MatchMaker handle new dynamically defined types?
  - What interface reference is provided for instantiable service, or service still in the process of design or implementation?
  - What is the type of the Policy Controller interface?


# How About ODP “Search” Operation?

- ***Input parameters:***
  - Service interface type
  - Search constraint (on links)
  - Matching criteria (on properties)
  - List of property names
- ***Output results:***
  - List of qualifying offers
    - Interface reference
    - List of property values
- ***Challenges:***
  - What is the type of this operation (given multiple types for properties)?
  - How does MatchMaker handle new dynamically defined types?
  - How are matching criteria specified for properties with complex data types?
  - May instantiate many servers, but use only one?
  - May want only portions of properties with complex data types?

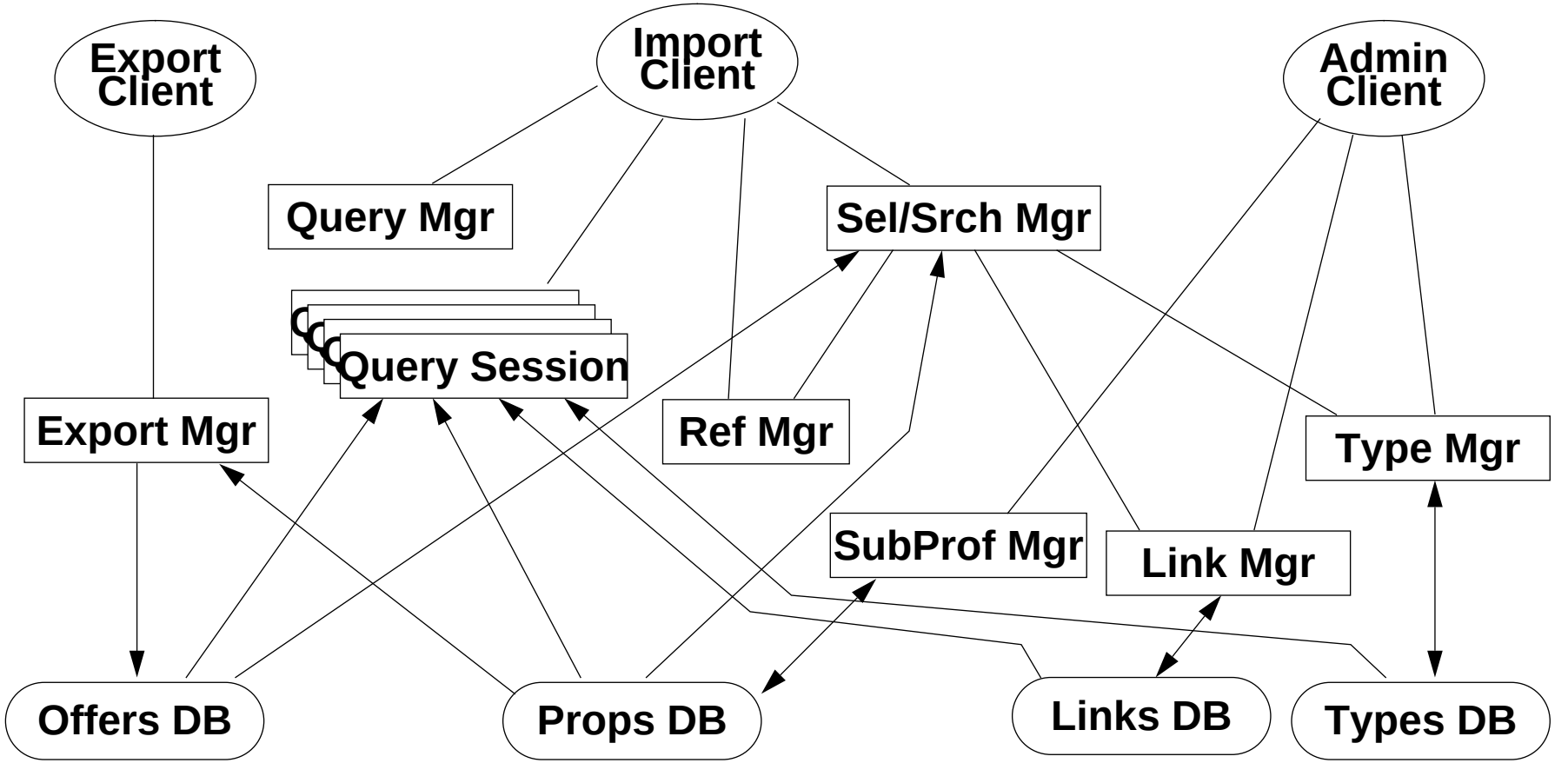

# Meeting the Challenges for MatchMaker Prototype

- *Implementation environment: Orbix + Allbase (CORBA + SQL).*
- *Support fairly complex data types for properties, with fair extensibility.*
- *“Register” property names and associated data types.*
- *Allow service interface reference to be missing or NULL.*
- *Provide SQL query facility.*
  - essentially gives matching criteria for complex property types
  - essentially allows retrieval of portions of complex properties
- *Support Select/Search operations, handling only “simple” property types.*
- *Do not instantiate services on Search operation; require separate GetRef.*
- *Only “simple” property types allowed as parameters for instantiable parameterized services; Policy Controller interface consists of Select operation; matching criteria must include OfferID = id.*


# MatchMaker Interfaces

- *Export manager*
- *Query manager*
- *Select/Search manager*
- *Subprofile manager - for registering property names*
- *Link manager - for federating MatchMakers*
- *Type manager - for finding compatible interface types*
- *Reference manager - for requesting instantiation of services*

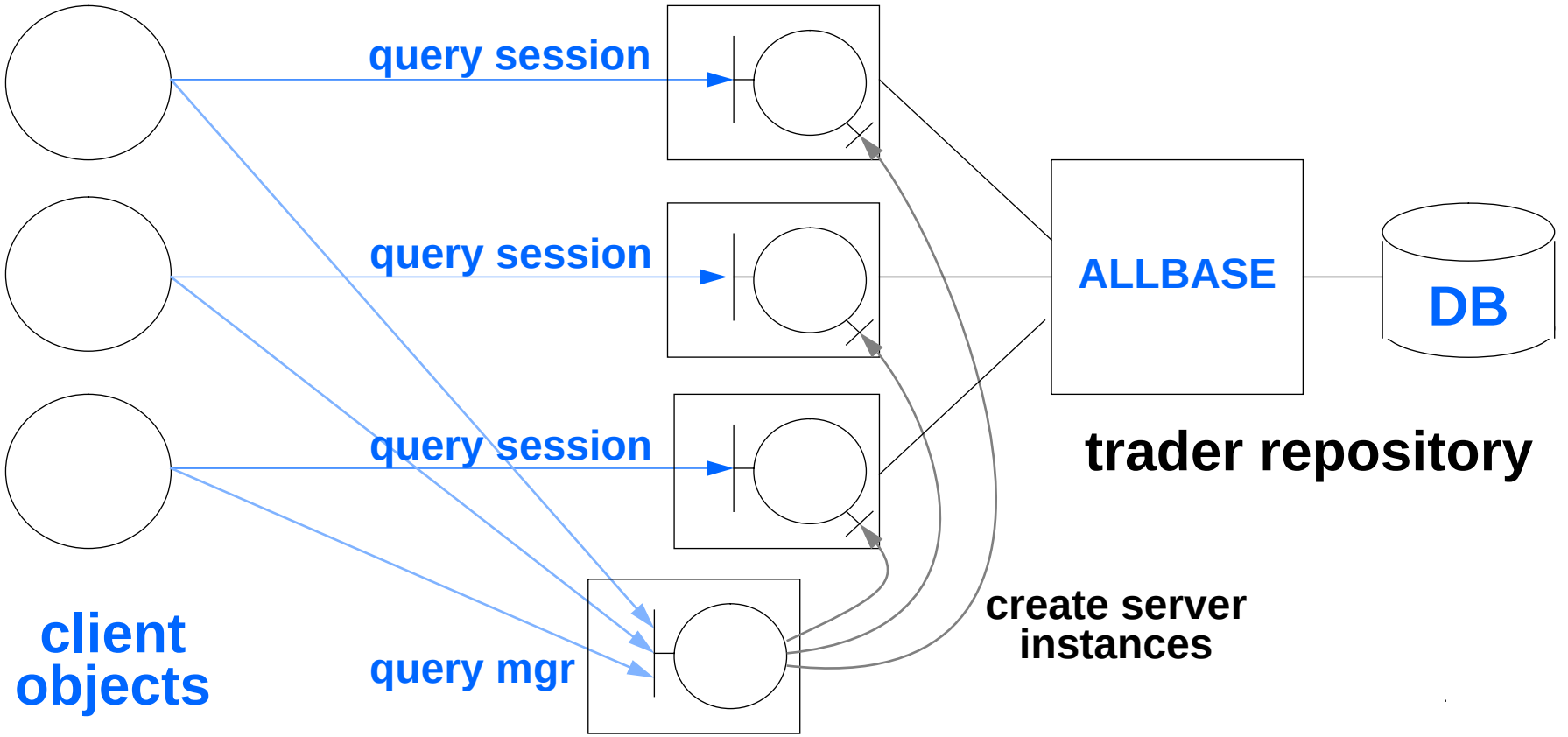

# Design Overview for MatchMaker




# Export Manager Features

- *Property value is represented as sequence < union {db_type} > where db_type includes short, long, float, double, string, octet_string, date, time, ... (SQL types).*
- *This can represent individual db_types, sequences of db_types, and structures whose elements are db_types.*
- *Multi-valued properties can be represented by multiple instances of <name, value> pairs with the same property name.*
- *This gives very good coverage of property data types observed in sample applications.*
- *Basic MatchMaker database supports properties whose values are single db_types, sequences of db_types, and multiple sequences of db_types.*
- *To support new structure, new table must be added to database, and an entry describing the structure must be added to a "property_types" table. This also allows properties whose values are sets/sequences of the structure.*

# Query Manager Features




# Select/Search Features

- ***Changes from ODP specification:***
  - **They only deal with “simple” properties.**
  - **Select returns all properties of offer.**
  - **Search constraint is named search policy, not predicate on link properties.**
  - **Search does not cause services/interceptors to be instantiated.**


# SubProfile Manager Features

- ***Add/delete/list:***
  - **Property data types**
  - **SubProfiles**
  - **Properties**


# Link Manager Properties

- *Add/delete/list links*
- *Given named search policy, produce ordered list of:*
  *< interface reference, search policy> pairs*
  *designating Select/Search interfaces to use and corresponding policies*


# Type Manager Features

- ***Add/delete/list offer interface types***
- ***Add compatibility assertions***

***Note that in the future compatibility should be computed from interface repository, not asserted.***


# Reference Manager Features

- ***Given OfferID and matching criteria, produce interface reference (typically requesting that service be instantiated) and list of properties of service.***


# Property/Offer Database Tables

- *Property type info - TypeName, TypeCode, Description, StorageTable*
- *Subprofile info - SPName, Description*
- *Property info - PropName, SPName, TypeName, Description*
- *Offer info - OfferID, ExporterID*
- *Offer-Property info - OfferID, SPName, PropName*
- *X-type property values - OfferID, SPName, PropName, ItemNum, PropValue (with PropValue replaced by multiple value columns in the case of a structure, one column for each field, and with an additional SeqNum column in the case of a sequence of sequences property data type)*


# Federating MatchMakers

- *Links provide mechanism for search among federation.*
- *Coordination of SubProfiles required for meaningful search.*


# Performance Characteristics of Prototype

- *No actual test data available yet.*
- *Launching servers is slow, response time after launching pretty good.*
- *Load modules of Orbix servers are large, especially when they also access Allbase.*


# Dependencies on Environment

- *Standard CORBA IDL, but some Orbix-specific calls for launching servers*
- *Standard ISO/ANSI SQL (some dynamic, some static)*
- *Some dependencies on specific versions of C and C++*
- *Selection of property data types to support definitely influenced by data types supported by SQL*


# Limitations

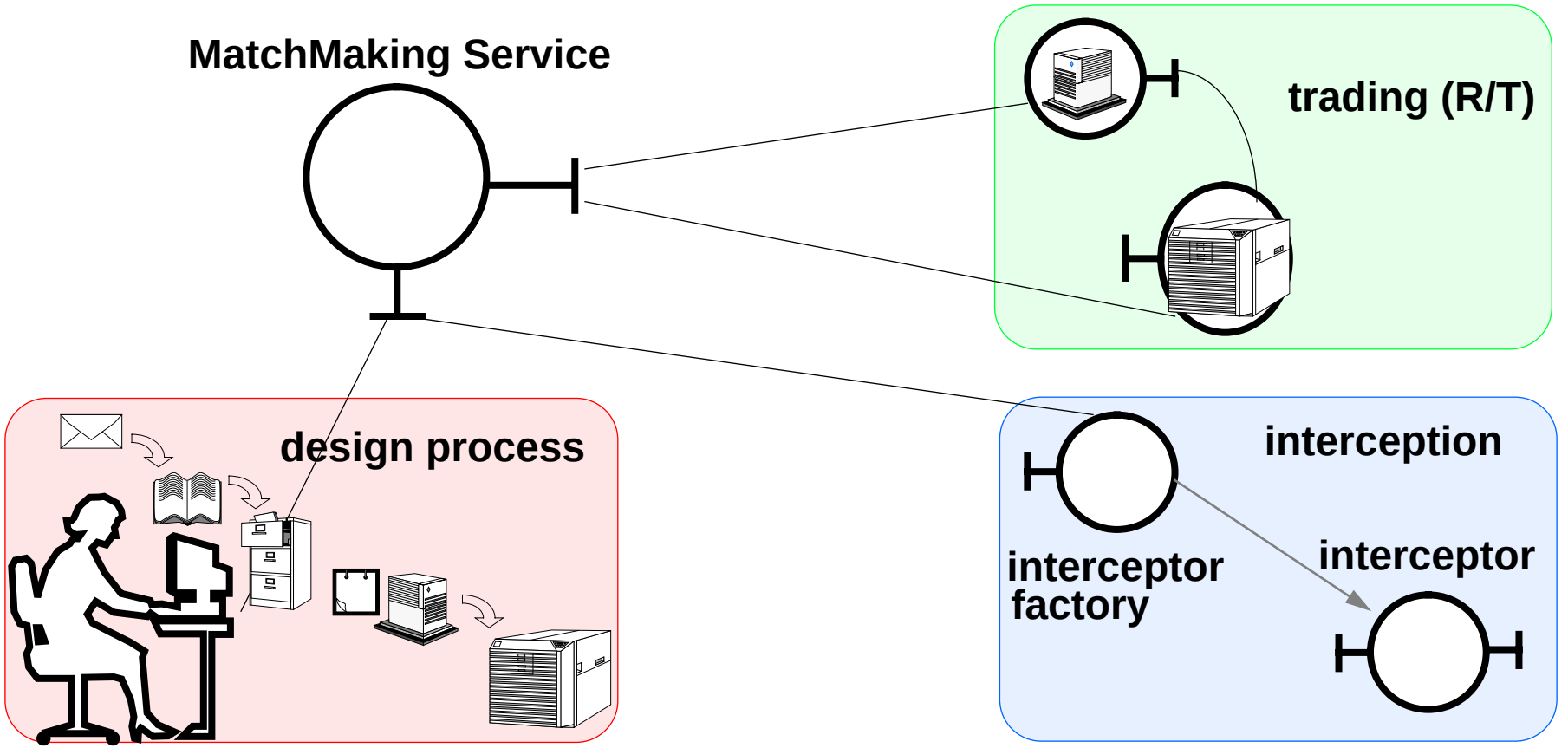
- ***Need better support for dynamic extensibility:***
  - **In C/C++ need to add data types dynamically at run time.**
  - **In CORBA need “dynamic skeleton interface”.**
  - **In SQL need ability to generate statement identifiers and cursor names dynamically.**


# Conclusions

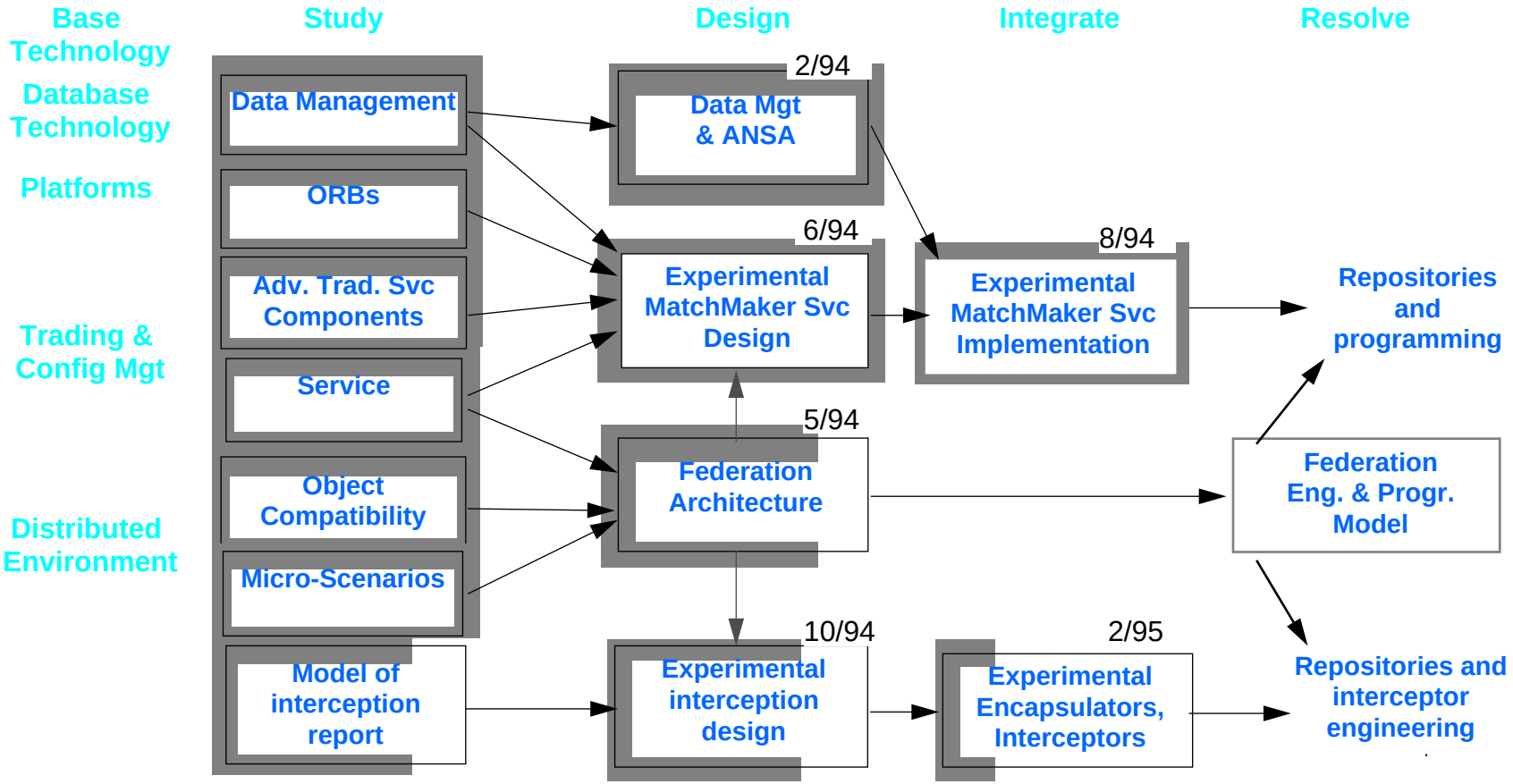
- *Prototype pretty much meets design goals.*
- *In some areas specification and implementation are not yet entirely stable.*
- *Extensibility of property data types is not optimal, largely because of limitations described in previous viewgraph.*
- *Client usability is not optimal, but can be improved by appropriate extensions to definitions of assignment operator and casts.*


# The way forward

# Use of MatchMaking Service supplied information



# Federation Plan: the way ahead




# Packaging the deliverable

- ***Documents***
  - **Briefing note: APM.1280**
  - **A designer's Introduction to Trading: APM.1140**
  - **An Information model for federation: APM.1229**
  - **Specification and design for MatchMaker: APM.1177**
- ***Prototype code***
  - **Available to sponsors**