The Zeus Agent Building Toolkit

ZEUS Methodology Documentation Part I

The Role Modelling Guide

Jaron Collis, jaron@info.bt.co.uk

Divine Ndumu, ndumudt@info.bt.co.uk

Applied Research and Technology, BT Labs

Release 1.01, August 1999

Index

Document History

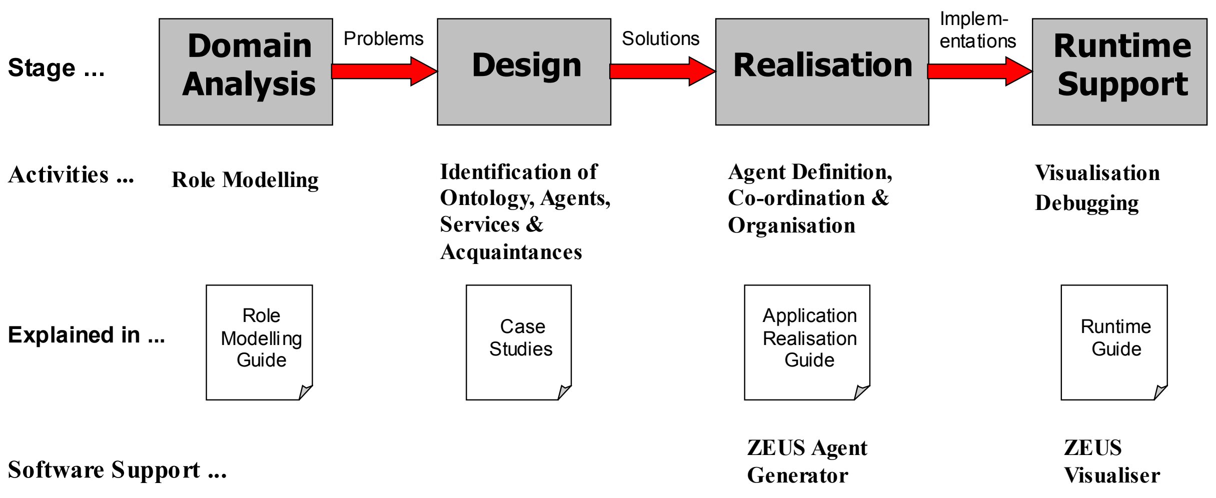
Version 1.01 - Added Section 3: Information Management Domain

1 INTRODUCTION

This document describes the first stage of the agent development methodology. In this context, the methodology is the set of methods and principles that drive the agent development process. Methodologies tend to be prescriptive in nature, not only facilitating the thinking and acting needed to perform the process, but also recommending methods and techniques that should be employed to achieve the process.

In building the ZEUS toolkit our intention has been to create a toolkit that can be applied to a wide variety of problems, and not just variations on a particular application. Thus the challenge for this documentation is to not only describe how to implement various agent applications, but also describe their general characteristics in such a way that future developers can use them as models for their own similar systems.

In common with most other structured development methodologies, the ZEUS approach consists of Analysis, Design and Realisation activities, as well as adding runtime support facilities that enable the developer to debug and analyse their implementations. The ZEUS methodology is summarised below:



1 - Domain Analysis

The purpose of the initial analysis stage is to model and understand the application problem. The ZEUS methodology does not explicitly prescribe any particular approach to problem analysis, leaving developers free to choose their own favourite approaches, like use cases or conceptual distances; the approach recommended is this document is Role Modelling, (see Section 1.2). As analysis techniques like role modelling are independent of means that will ultimately implement them, the role models discussed in this document should be realisable with any agent building system, and not just ZEUS. ZEUS does not currently offer any software support at this stage.

2 - Agent Design

By the time the design process begins the developer should know what agents will be present, and what responsibilities they will fulfil. Hence this stage involves the translation of role responsibilities into the agent-level problems they represent, and deriving appropriate solutions. Where the analysis process involved understanding the problem requirements, the design process involves expertise, knowing when and how to reuse and adapt existing proven solutions. As this knowledge is difficult to accumulate, each of the role models comes with an associated Case Study that describes the reasoning behind a design for a sample application. There is no ZEUS software support at this stage.

3 - Agent Realisation

The objective of this process is to realise working agent implementations from the conceptual designs created during the previous stage. The agent realisation process consists of several stages, which are closely coupled to the levels of abstraction that exist within a ZEUS agent. This stage is where ZEUS begins to offer software support, providing an Agent Generator tool through which the designs can be entered, and then be used to generate Java source code for the agents.

4 - Runtime Support

The ZEUS approach does not end with the creation of the agents; there is also a suite of runtime support tools that are available through the Visualiser agent. This reflects the fact that the development process is unlikely to have ended with the implementation of the agents, as they still need to be tested, debugged and optimised.

The focus of this document is the first Problem Analysis stage, using an approach called Role Modelling; this approach is outlined in the next section. The other stages of the methodology are explained in their own documents.

1.1 An Introduction to Role Modelling

Agent roles and role models provide a vocabulary for describing agent systems, with each role describing a position and a set of responsibilities within a certain context or role model. This approach encourages developers to think of the problem in terms of the roles that need to be played, and the responsibilities associated with each role.

The role models in this document were inspired by the work of Kendall, [1], which formalises the definition of an agent role so that it can be modelled, designed, and implemented in software. This work is effectively an agent-oriented extension of role modelling, as practised in conventional object oriented software engineering, [2]. In fact, the qualifying criteria for roles are very similar those of objects, i.e. role should:

- *be modular:* a role describes a set of entities that can occupy the same position in a reoccurring structure, hence the role must be modular so that new "players" can be assigned without any impact on the rest of the role model
- *have high cohesion*: to promote modularity a role must have a well-defined set of related responsibilities and a clearly defined function; the responsibilities must form a cohesive unit
- *be parsimonious*: a role should not have extraneous responsibilities
- *be complete*: a role should not be trivial; trivial roles should be merged with other roles
- *have low coupling*: dependencies between roles must be minimised

Role modelling is relevant to most facets of the agent development lifecycle, addressing the specification, analysis, design, implementation, and maintenance of agents. For instance, problem specification is aided by the provision of role models with concepts and abstractions that describe the agent requirements. Role models are also patterns of interaction, providing a readily comprehensible means of analysing the problem in question. This document contains many such agent patterns that should be checked for relevance during agent system analysis.

Although role modelling is primarily associated with the specification and analysis of a problem, its influence extends to the design stage. Typically an agent will play many roles, and so during design the previously identified roles will be composed together into agent entities that can support all of the relevant roles. Finally the functionality associated with each role can be implemented.

How to use this Documentation

The role models in this document are grouped into **domains**. These domains have been chosen after analysing existing agent applications and grouping together those that address similar problems or exhibit similar behaviour. The domains provide a context that enables developers to compare their planned system with existing applications.

Thus each section the subject domain possesses several **role models**. Role models are 'architectural patterns' that depict the high-level similarities between related systems, i.e. the *problems* inherent to each domain, but not how they were *solved*.

So, next the reader should consult the appropriate case study document, these describe the problems inherent to particular role models, potential solutions and how they can be *realised* (in this case, using the ZEUS tool-kit). This can be summarised as the following progression:

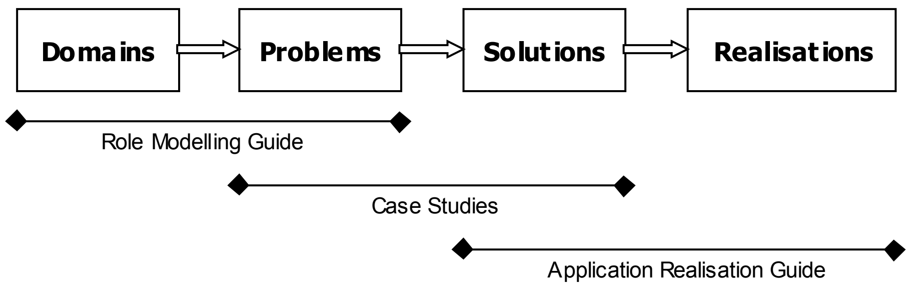


Figure 1.1: The phases of the ZEUS Agent Creation Process, and the documents that describe them

The Components of a Role Model

The role models in this document are described from several perspectives. Each role model entry begins with a description summarising its main features and general applicability. The model's constituent roles are then shown in a Role Model diagram, like that shown in Figure 1.2.

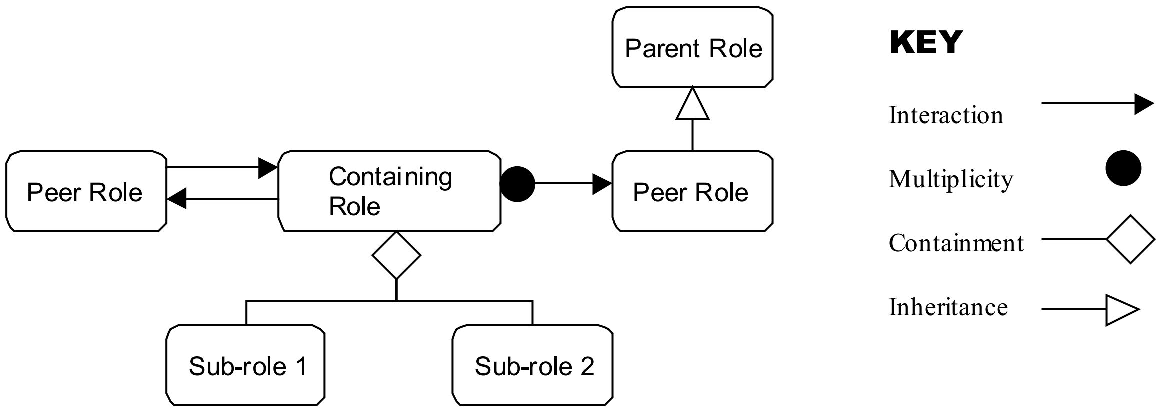


Figure 1.2: The Notation used within a typical Role Diagram

Role model notation originates from the UML (Unified Modelling Language) class diagram notation, [3]. In role diagrams the key concepts are roles rather than classes, (represented by rectangles), whilst containment and inheritance are depicted in the same fashion as in class diagrams, with diamond and triangle headed lines respectively.

The only difference in notation arises from the key difference between class diagrams and role diagrams. Whereas class diagrams describe the static relationships between classes, role models describe the dynamic interactions between roles. Hence the UML class diagram has been augmented with additional notation to depict interactions: the arrowhead line. Where an arrowhead line is shown with a filled circle, this means more than one simultaneous interaction can occur between entities playing these roles.

After the role diagram, the next component of each role model entry is the *collaboration diagram*. This abstracts away from the specialisation and containment relationships between roles and instead concentrates on how they interact, an example is shown in Figure 1.3.

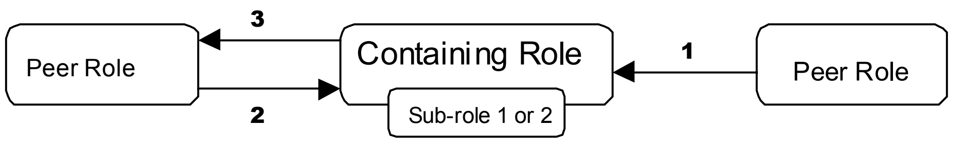


Figure 1.3: The Notation used within a typical Role Diagram

The main difference between collaboration and role diagrams is that only interactions are shown. Each interaction is annotated with a number, which refers to an explanation of the interaction found later in the role model. Where the collaboration diagram is made easier to understand by the inclusion of sub-roles, these are shown inside their containing role.

The next part of a role model entry is the role description section; this consists of tables like that shown in Figure 1.4. The purpose of this section is to describe each role in terms of its social obligations and application-specific functionality. Hence each of the interactions shown in the collaboration diagram will appear in the corresponding role description.

As well as interactions with other roles, each entry also describes the interactions between the role and its external interfaces. The 'external interfaces' represent the means through which the role performs its application-specific activities, such as accessing databases, or reading information from a user interface. Some entries may also describe the social abilities necessary for the role to fulfil its social responsibilities.

ROLE NAME	
Role Model: the model that contains this role	
Relationships to other roles: roles that this contains or specialises	
Description: A written description of this role's characteristics, abilities, responsibilities etc.	
Responsibilities:	**Collaborators:**
[1] An outgoing interaction	⇒ Source Role
[2, 3] An incoming and outgoing interaction	⇐ ⇒ Source and Destination Role
External Interfaces:	
An application specific activity	
Necessary Social Protocols: The abilities necessary to fulfil its responsibilities	

Figure 1.4: The component fields of a Role Description

Role models are patterns, representing the simplest possible solution for a particular application. As the role model is effectively the lowest common denominator solution the final part of each role model describes common variations that extend its functionality.

1.2 Assigning Roles to Agents

By the end of the analysis stage the developer will have identified the roles that will need to be played to realise the application. Consequently the first activity of the design stage is to allocate these roles to agents. This raises the question: what is an agent? Papers on this subject acknowledge that this is not an easy question to answer, [4,5], and these are recommended reading for those new to the agent field.

Of course a methodological development process must provide more concrete guidance, what is needed is some means of setting the granularity at which the domain is modelled. Hence in this section two metrics are described that may prove useful in determining whether candidate entities would make appropriate agents:

1) The Sphere of Responsibility Test

2) The Point of Interaction Test

The Sphere of Responsibility Test

The first metric is derived from the fact that agents should be autonomous, i.e. be responsible for the control of resources and provision of services. This area of control is known as the agent's *Sphere of Responsibility*. Thus when considering what agents will exist the developer will need to consider how the application domain will be partitioned, i.e. the degree of coupling between roles.

This principle is best illustrated with an example. Consider a trading scenario based on the Role Model in Section 3.1, this involves the following roles:

- Entry of user preferences
- Negotiation with other traders
- Provision of trading expertise
- Payment and ownership exchange

One solution would be to create an agent for each role, i.e. an agent to interact with the user, another to interact with other agents, a third to provide trading expertise to those who need it and a fourth to facilitate the transaction. Whether this is an appropriate solution depends entirely on the nature of the application. Just because agents provide a means of distributing an application does not mean that activities should be distributed as widely as possible. In fact it is often the case that it is more efficient to centralise certain activities.

Returning to our example scenario: suppose the user interface is to run on a computer in a trader's office, and negotiations will be performed by another piece of software on behalf of all traders in the company. In this scenario there would seem to be two different areas of responsibility: a sphere where activities local to individual traders are supported, and a company-wide sphere supporting activities that are more efficiently centralised. Hence the Sphere of Responsibility test is simply:

“Each sphere of responsibility should possess a single agent”

Using this rule of thumb to partition our example scenario would result in the arrangement illustrated in Figure 1.5, this assumes the presence of 4 trader agents, each with their own local sphere of responsibility. Each trader agent will use the local computing resources to provide personalised user interaction and trading expertise. The negotiation and settlement activities have been centralised and placed in the company sphere, where they will be available to the whole company. However, it is important to stress that there is no 'correct' solution, rather the most appropriate arrangement depends on the application and is at the discretion of the developer.

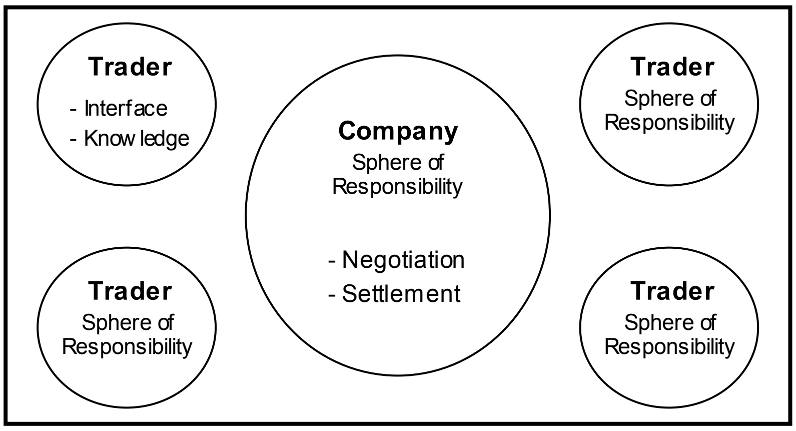


Figure 1.5: The Partitioning of an application into Spheres of Responsibility

It is also worth noting another factor that may help identify candidate agents. Agents tend to be *responsive*, i.e. able to perceive their environment and respond accordingly to events that affect their own sphere of responsibility.

The Point of Interaction Test

The second metric is related to the Sphere of Responsibility test, but extends it by considering the social dimension of agents. Agents are often distinguished from other software systems by their ability to interact intelligently and constructively with other agents and people. Hence in an agent application resources and services may not be directly accessible, but invoked by requesting the agent responsible for their control.

The purpose of this test is to help separate application resources from the entities that will use them to provide services. This is particularly relevant to applications where agent systems serve as the interfaces to legacy systems such as databases. For instance, consider a warehouse supply scenario with a stock inventory and several delivery vans.

To fulfil an order the inventory must be queried and availability of delivery verified. Who then would we interact with in order to place an order? We could query the stock inventory directory, but that seems unrealistic, companies rarely make their databases publicly readable. It is more likely that queries would be sent to the inventory's owner, the warehouse, which would access the inventory and provide a response (perhaps using additional knowledge like when new stocks are due).

It is also unlikely that customers would contact the delivery vans directly, indeed as non-software resources it is unlikely that the vans will feature at all in our solution. Instead, we shall assume that vans take their instructions from a delivery schedule that is maintained by a delivery company that has a contract to deliver goods from the warehouse.

This example serves to illustrate the difference between resources and agents: agents affect resources, interactions affect agents. This provides us with the ‘Point of Interaction’ test, which states:

“The access point for information, expertise and services is a good agent candidate”

Using this rule of thumb to partition our example scenario would result in the arrangement illustrated in Figure 1.6, this shows how the Point of Interaction test has been used to justify the existence of two agents in our example application. One is the Warehouse, which is the interaction point for stock queries and orders, and the other is the Delivery Company, the interaction point for requesting and querying deliveries.

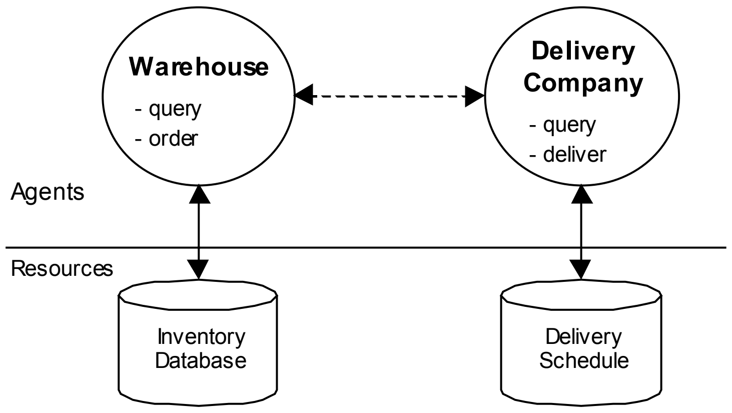


Figure 1.6: The Partitioning of an application into Agent and Resource levels

Notice how there is no mention of vans - only the software based delivery schedule; this is a significant principle: choose resources that can readily interact with agents. Again the most appropriate arrangement will depend on the application, and is at the discretion of the developer.

Combining Roles

The preceding examples have illustrated how individual roles do not necessarily need to be played by individual agents. In fact it is more likely that several roles will be combined and be performed by a single agent. Roles may combine in various ways:

- the behaviour may just add together, (as is the case for independent behaviours)
- some behaviour may override other behaviour, (this will tend to happen if one role specialises the behaviour of another)
- the behaviour might be combined synergistically, (this will occur when the roles are neither mutually exclusive nor independent, here the roles will typically complement and enhance each other).

The examples in the Case Studies will illustrate how to combine roles appropriately.

2 THE ZEUS APPLICATION ROLE MODEL

The remaining sections of this document describe role models, beginning with the ZEUS application role model. This describes the functionality inherent to every agent application created with the ZEUS tool-kit. As the roles in this role model are basic to multi-agent applications, they are not repeated in the role models in future sections, where they are considered implicit.

Abstract

The Zeus tool-kit provides four default roles that implement the functionality inherent in multi-agent applications. Agents in the Task Agent role will perform the application's domain-specific activities. These will interact with the Name Server, who maintains a registry of known agents, and the Directory Facilitator, which maintains a record of agent abilities. The Visualiser provides a graphical depiction of agent activity that is independent of the application.

ZEUS Application Role Model Diagram

For clarity the ZEUS application role model is presented from two perspectives. Figure 2.1 illustrates the Utility agent roles; these roles will be present in nearly all applications, providing the support services necessary for the society to function.

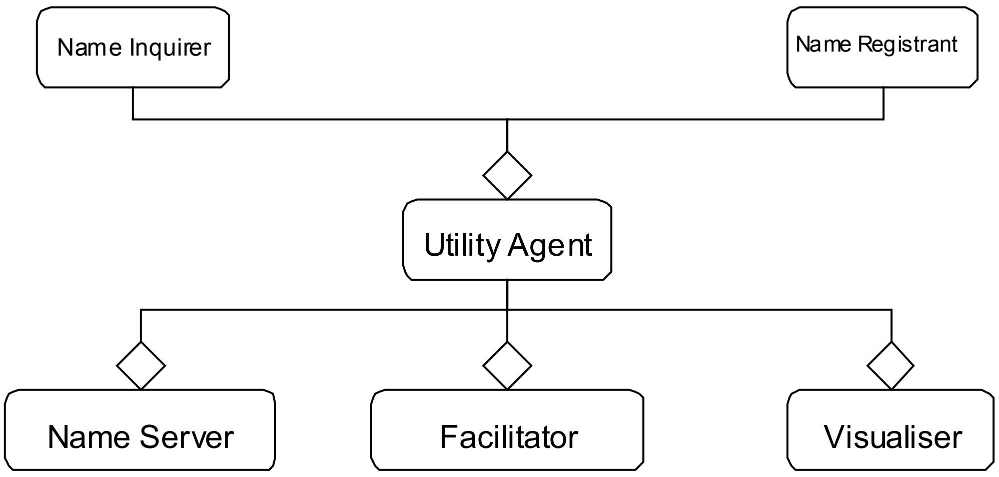


Figure 2.1. The default roles present within the Zeus Toolkit, shown relative to the Utility Agent role

The interpretation of Figure 2.1 is straightforward: the utility agent role (and thus the Name Inquirer and Name Registrant roles) is contained by each of the Name Server, Facilitator and Visualiser. The implication of every utility agent playing the Name Inquirer and Name Registrant roles can be found in the Role Descriptions that follow: namely that each utility agent is compelled to register itself with a Name Server, and thereafter can query the Name Server for the addresses of others.

Figure 2.2 provides a different perspective on the ZEUS application role model, showing the relationships between the utility agents and all agents that play the task agent role. The interactions between these roles are then explained in Figure 2.3.

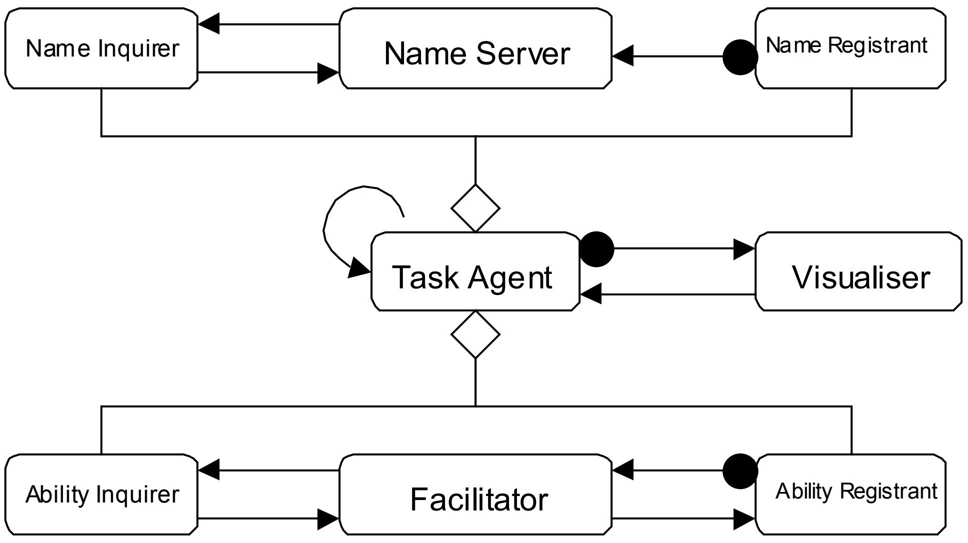
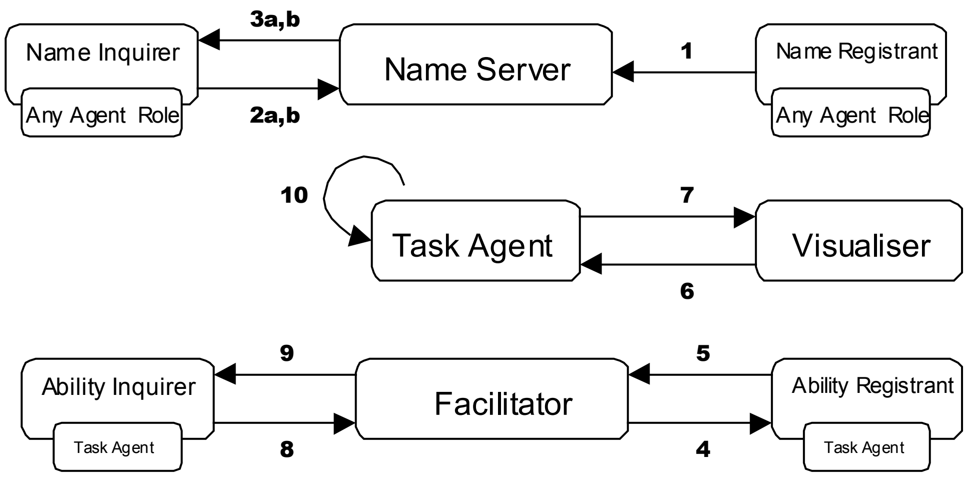


Figure 2.2. The default roles present within the Zeus Toolkit shown relative to the Task Agent

Collaboration Diagram for Zeus Application



Interaction Summary:

	Collaboration	**Explanation**
1	Registration	Agents notify the Name Server of their presence
2a	'Resolve' Query	A request for the network location of a named agent
2b	'List' Query	A request for all agents of a particular type
3a	Location Response	The location of the agent previously in question
3b	List Response	The list of agents previously in question
4	Ability Request	Asks for information about recipient's abilities
5	Ability Response	Information about an agent's current abilities
6	Inform Request	Asks all activity be forwarded
7	Activity Notification	A copy of any message sent
8	Find Request	Asks for agents with particular abilities
9	Find Response	A list of agents matching the desired criteria
10	Inter-Agent Message	Any other message

Zeus Application Role Descriptions

NAME SERVER	
Role Model: Zeus Application	
Relationships to Other Roles: Contains `Utility Agent`	
Description: An analogous role to that played by Internet Domain Name Servers, this maintains a registry of known agents and their actual network locations. This information is typically accessed in one of two ways: by requesting the address of a named agent, or by requesting the identities of all known agents. The latter tends to be used by Utility Agents only (cf.).	
Responsibilities:	**Collaborators:**
To maintain society's synchronisation clock	
To maintain directory of agents and their locations	
[1] To receive and process registration messages	⇒ Name Registrants
[3a] To provide network locations of named agents	⇐ Name Inquirers
[3b] To provide the list of known agents	⇐ Utility Agents
Prerequisites:	
Social protocols for receiving registrations and queries, and notifying results	
Implementation: Provided as part of the ZEUS toolkit	

NAME REGISTRANT	
Role Model: Zeus Application	
Responsibilities:	**Collaborators:**
[1] To register its presence on start-up	⇒ Name Server
Prerequisites:	
Social Protocols to enable registration	
Implementation: None necessary, this role is encoded into all agents created using ZEUS	

NAME INQUIRER	
Role Model: Zeus Application	
Responsibilities:	**Collaborators:**
[2a, 3a] To request location of a named agent	⇐ ⇒ Name Server
Prerequisites:	
Social protocols for sending queries and receiving results	
Implementation: None necessary, this role is encoded into all agents created using ZEUS	

UTILITY AGENT	
Role Model: Zeus Application	
Relationships to Other Roles: Contains `Name Registrant, Name Inquirer`	
Responsibilities:	**Collaborators:**
[2b, 3b] To request the list of known agents	⇐ ⇒ Name Server

FACILITATOR	
Role Model: Zeus Application	
Relationships to Other Roles: Contains `Utility Agent`	
Description: An analogous role to that played by Internet service directories such as yahoo.com. Facilitators maintain a registry of agents and the services they are known to offer. On start-up a Facilitator will obtain the list of active agents from a name server, and then collect service information by periodically polling each one. An agent needing to find another will then be able to discover an appropriate source by querying the facilitator.	
Responsibilities:	**Collaborators:**
To maintain directory of agents and their abilities	
To possess and understand the application ontology	
[4,5] To send and receive ability queries	⇐ ⇒ Ability Registrants
[8, 9] To answer questions on abilities of known agents	⇐ ⇒ Ability Inquirers
Prerequisites:	
Social protocols for receiving registrations and queries, and notifying results	
Implementation: Provided as part of the ZEUS toolkit	

ABILITY REGISTRANT	
Role Model: Zeus Application	
Responsibilities:	**Collaborators:**
[5] To register its abilities	⇒ Facilitator
Prerequisites:	
Social Protocols to enable registration	
Implementation: None necessary, this role is encoded into all agents created using ZEUS	

ABILITY INQUIRER

Role Model: Zeus Application	
Responsibilities:	**Collaborators:**
[8,9] To ask for identities of agents with requisite abilities	⇐⇒ Facilitator
Prerequisites:	
Social protocols for sending queries and receiving results	
Implementation: None necessary, this role is encoded into all agents created using ZEUS	

VISUALISER

Role Model: Zeus Application	
Relationships to Other Roles: Contains `Utility Agent`	
Description: Provides an application-independent visualisation service. On start-up a Visualiser will obtain the list of active agents from a name server, and then send messages to each requesting that they forward a copy of any future message sent. When received these copied messages are interpreted and used to update the Visualiser's model of the agent society, which can be viewed in real-time from various perspectives.	
Responsibilities:	**Collaborators:**
To visually depict aspects of the agent society	
[6] To request activity reports	⇒ Zeus Agents, Utility Agents
[7] To receive and interpret activity reports	⇐ Zeus Agents, Utility Agents
Prerequisites:	
Social protocols for receiving activity messages and queries	
Implementation: Provided as part of the ZEUS toolkit	

TASK AGENT

Role Model: Zeus Application	
Relationships to Other Roles: Contains `Name Registrant, Name Inquirer, Ability Registrant, Ability Inquirer`	
Description: This role encapsulates the behaviour inherited by every 'task' agent created using the Zeus toolkit. In contrast to the agents that play 'utility agent' roles, those playing 'task agent' roles will be responsible for some domain-specific activity.	
Responsibilities:	**Collaborators:**
To perform designated activities	
To load and use an agreed ontology	
External Interfaces:	
To implement domain-specific activities	
Prerequisites:	
Social protocols for registering, querying and sending and receiving task-related messages	
Implementation: Application dependent, see subsequent role models	

Task Agent Specialisation

It should be stressed that a task agent possesses very little useful functionality by default. Consequently task agents are provided as the basis for more specialised roles, these are documented in the subsequent sections of this document.

3 THE INFORMATION MANAGEMENT DOMAIN

Constituent Role Models

1) The Shared Information Space

Domain Characteristics

- Applications in this domain are characterised by asymmetrical possession of information; i.e. some entities will possess information that others do not, and so the entities must communicate to share information necessary to their continued operation.

Useful Links

To follow.

ROLE MODEL

3.1 The Shared Information Space

Abstract

A shared information space involves two types of entity, Publisher and Subscriber. The publisher maintains some repository of information to which it has sole access: this is consistent with the principle of agent autonomy. The Subscriber entities require the information owned by the Publisher, which they obtain by sending messages. This role model assumes there is only one Publisher, with a canonical repository of information. There may however be several subscribers, and they may use the same source of information for their own individual purposes.

The Publisher contains 3 sub-roles: a Model role is responsible for maintaining its stored information, a View role is responsible for visualising it, and a Controller role serves as an intermediary between the stored information and the Subscribers. As the names suggest, these roles have been inspired by the classic Model-View-Controller pattern. The complete role model diagram is shown in Figure SIS1.1

As Exemplified by:

"To Follow"
"To Follow"

Similarities and Differences

This role model uses the name server role from the `Zeus Application` role model to provide network independence. Although the identity of resource producers could be encoded into the agents that will use them, a facilitator is recommended for flexibility. Both roles are omitted from figure SIS1.1 for clarity.

Shared Information Space Role Model Diagram

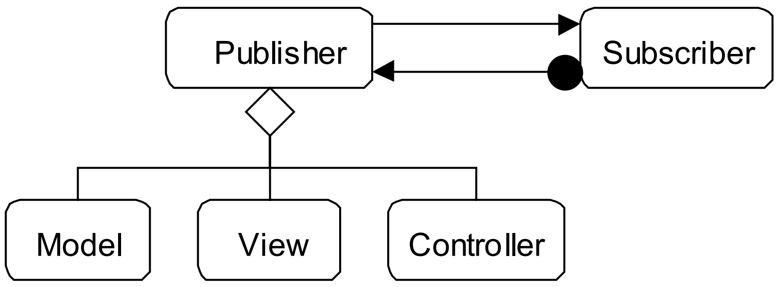


Figure SIS1.1: The roles present within a Simple Supply Chain

Figure SIS1.1 shows how the Publisher role is in fact comprised of three sub-roles, whilst no assumptions are made as to the structure of the Subscriber role. It also shows that there is assumed to be only one Publisher, but that this can service multiple Subscribers. The interactions involved are considered next.

Shared Information Space Collaboration Diagrams

This role model contains three sets of interactions; there is interaction between the Controller role of the publisher and the Subscriber, which involves registration and the issue and response to queries. Responses to queries are then formulated by consulting the Model. The View role can also access the model, and is shown in Figure SIS1.2 as performing the same interactions as the Controller. Readers familiar with the Model-View-Controller design pattern will recognise this arrangement, as it is common to see the View and Controller are often implemented as a single component.

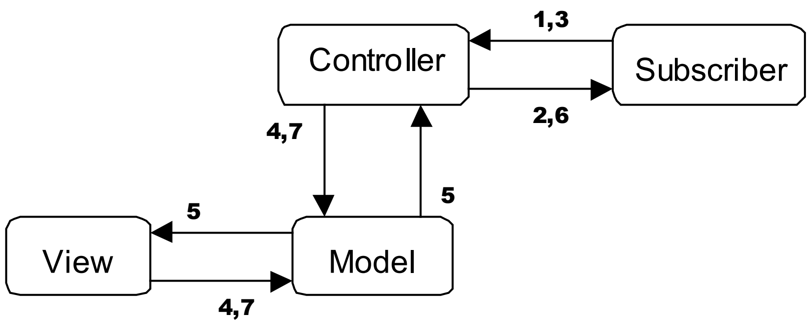


Figure SIS1.2: The Interactions between the roles of the Shared Information Space

Interaction Summary:

	Collaboration	Explanation
1	Registration	Subscribers subscribe to the Publisher
2	Registration Response	Publisher notifies Subscriber
3	Information Requested	A request for information is submitted
4	Model Queried	Model is queried to satisfy query
5	Model Response	Information requested is returned to Controller
6	Information Response	Controller presents information to Subscriber
7	Model Updated	A change is made to the information of the model

Shared Information Space Role Descriptions

CONTROLLER	
Role Model: Shared Information Space	
Relationships to other roles: Contained by the `Publisher` role	
Description: This role serves as an information provider and is thus the pivotal role of the Publisher. This handles incoming subscription and information requests from Subscribers by adapting their requests, querying the model and providing a response.	
Responsibilities:	**Collaborators:**
[1, 2] To listen for and reply to subscription requests	⇐ ⇒ Subscribers
[3, 6] To listen for and reply to information requests	⇐ ⇒ Subscribers
External Interfaces:	
To query model in response to subscriber's information requests	
Prerequisites:	
None assumed	

MODEL	
Role Model: Shared Information Space	
Relationships to other Roles: Contained by the `Publisher` role	
Description: This role stores the information held by the Publisher entity, and provides its sister roles with a means of accessing and changing the model.	
Responsibilities:	**Collaborators:**
[4, 5] To provide information on demand	⇐ ⇒ Controller, View
[7] To enable changes to be made to stored information	⇐ Controller, View
External Interfaces:	
To encapsulate a repository of information	
Prerequisites:	
None assumed	

VIEW	
Role Model: Shared Information Space	
Relationships to other roles: Contained by the `Publisher` role	
Description: This Publisher role is optional; it is typically implemented when it is necessary to depict the information stored by the model. There can be multiple instances of this role according to the number of different ways the information is displayed.	
Responsibilities:	**Collaborators:**
[4, 5] To query and represent the model	⇐ ⇒ Model
[7] To effect changes in the model	⇒ Model
External Interfaces:	
To visualise the contents of the model	
Prerequisites:	
None assumed	

SUBSCRIBER	
Role Model: Shared Information Space	
Description: This role stores the information held by the Publisher entity, and provides its sister roles with a means of accessing and changing the model.	
Responsibilities:	**Collaborators:**
[1, 2] To make and receive subscription requests	⇐ ⇒ Controller
[3, 6] To make and receive information requests	⇐ ⇒ Controller
External Interfaces:	
To perform its application specific activities	
Prerequisites:	
None assumed	

4 THE MULTI-AGENT TRADING DOMAIN

Constituent Role Models

1) Distributed Marketplace

2) Institutional Auction

Domain Characteristics

- Applications in this domain typically possess many agents whose primary motivation for interaction is trade.
- The subject of trade may vary, from tangible items like books to intangible commodities like network bandwidth.
- Prerequisites for inter-agent trade are compatible interaction protocols and a common trading ontology.

Useful Links

- http://ecommerce.media.mit.edu/
- http://botspot.com/s-comm.htm
- http://ecom.infm.ulst.ac.uk/Research_Groups_Centres.html
- http://www.yahoo.com/Business/Electronic_Commerce/

4.1 The Distributed Marketplace

ROLE MODEL

Abstract

This role model illustrates the archetypal decentralised marketplace. Such a market is characterised by an absence of institutional rules and centralised arbiters. Instead trading occurs directly between buyers and sellers using a protocol agreed between the parties concerned. It is assumed that the roles of buyer and seller are not mutually exclusive, i.e. agents may be buyers and sellers simultaneously.

The decentralised nature of this type of marketplace means potential buyers require a means of finding vendors, and prospective vendors require a means of advertising. In static marketplaces, (i.e. where membership does not change), it may be sufficient to equip each agent a priori with knowledge of its peers. A more flexible alternative, and one that is used by dynamic marketplaces, is to create at least one agent to play the role of a broker. The broker maintains an up-to-date registry of prospective buyers and sellers, providing an efficient means for agents to locate appropriate transaction partners.

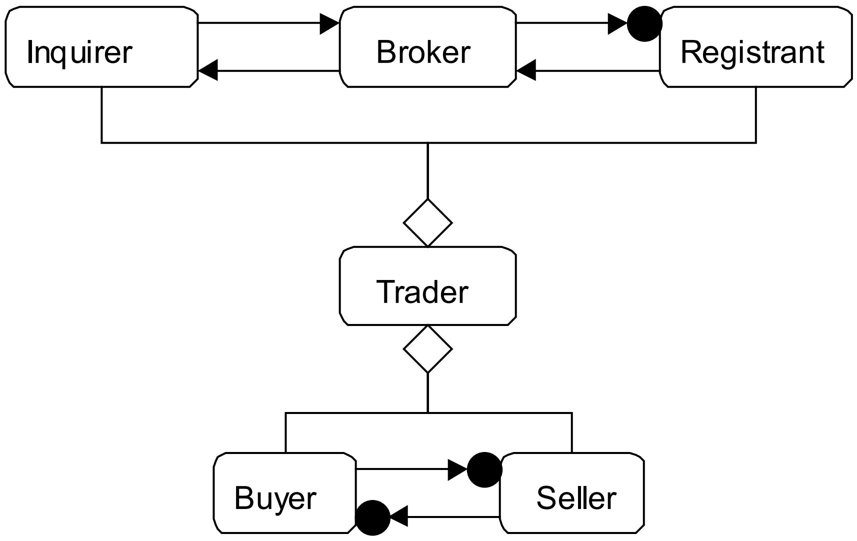
It is also worth noting that this role model does not assume the presence of a trusted third party that mediates transactions and transfers ownership of the traded goods. Consequently this role model is more applicable to scenarios where each party trusts the other to deliver on its agreements. Where this can not be guaranteed it may be sensible to extend this model with a Mediator, (q.v. Common Variations).

As Exemplified by:

KASBAH	"A Real-Life Experiment in Creating an Agent Marketplace": Chavez et al, Proceedings of PAAM '97, p159-178. http://ecommerce.media.mit.edu/
FERMAT	"Building Electronic Marketplaces with the ZEUS Toolkit": Collis and Lee, Agent Mediated Electronic Trading Workshop 1998, p17-32. http://www.labs.bt.com/projects/agents/
SICS MarketSpace	"An Agent-based Market Infrastructure": Eriksson et al, Proceedings of 2nd Agent Mediated Electronic Trading Workshop 1998, p33-48. http://www.sics.se/~market/

Similarities and Differences

Implicit in this role model is the presence of a name server, as described in the `Zeus Application` role model; although this is not shown in Figure DM1.1 for clarity. The `Distributed Marketplace` role model expands the role of the Zeus Application's Facilitator role, replacing it with a role termed 'Broker'.



Distributed Marketplace Role Model Diagram

Figure DM1.1. The roles present in a typical distributed marketplace application.

Distributed Marketplace Collaboration Diagram

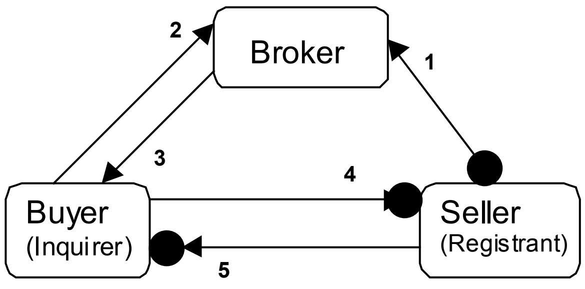


Figure DM1.2. The interactions between the roles of a typical distributed marketplace.

Interaction Summary:

	Collaboration	**Explanation**
1	Registration	Sellers register or de-register their items for sale
2	Find Request	Asks for agents selling items of interest
3	Find Response	A list of agents matching the desired criteria
4	Buyer Offer	Message containing buyer's proposal
5	Seller Response	Seller's reply to a previously submitted offer

(The initial interactions between agents as they register with a Name Server are not shown)

Distributed Marketplace Role Descriptions

BUYER	
Role Model: Distributed Marketplace	
Relationships to other roles: Contained by the `Trader` role	
Description: This is the role played by potential purchasers. No assumptions are made as to the expertise of the buyer, which may be a simple proxy or an intelligent trader with its own buying strategies. Note how the Buyer is not required to register itself with a Broker. The knowledge held by a Buyer will vary. They must be aware of the application ontology (the set of tradable concepts and their attributes), and will probably possess expertise on pricing and trading strategies.	
Responsibilities:	**Collaborators:**
[2, 3] To request information on known sellers	⇐ ⇒ Brokers
[4, 5] To communicate bids to potential vendors	⇐ ⇒ Sellers
External Interfaces:	
To facilitate the entry of user transactions	
To interpret vendor responses	
To facilitate payment and ownership transfer	
Prerequisites:	
At least one trading or auction protocol	

SELLER	
Role Model: Distributed Marketplace	
Relationships to other Roles: Contained by the `Trader` role	
Description: This is the role played by potential vendors. No assumptions are made as to the expertise of the seller, which may be a simple proxy or an intelligent trader with its own selling strategies. A key issue for Sellers is advertising, this may necessitate registering with a Broker if they are to be found by potential buyers. The knowledge held by a Seller will vary. They must be aware of the application ontology (the set of tradable concepts and their attributes), and will probably possess expertise on pricing and trading strategies.	
Responsibilities:	**Collaborators:**
[1] To advertise new items for sale	⇒ Brokers
[4, 5] To receive and respond to bids	⇐ ⇒ Buyers
External Interfaces:	
To facilitate user selling preferences	
To interpret bids	
To facilitate payment and ownership transfer	

Prerequisites:
At least one trading or auction protocol

TRADER
Role Model: Distributed Marketplace
Relationships to other Roles: specialises `Task Agent`, contains `Buyer` and `Seller`
Implementation: Modified Task Agent; see Case Study #1

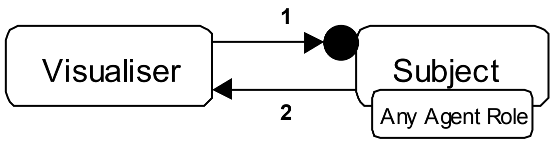
BROKER	
Role Model: Distributed Marketplace	
Relationships to other Roles: A variant of `Facilitator`	
Description: A variation on the standard Zeus Application Facilitator role which, instead of maintaining a registry of abilities, stores notices about goods sought and for sale. Brokers may be reactive (where active traders will advertise themselves) or proactive (where the broker will request information on each trader's status). Like the Facilitator the knowledge collected by the Broker is distributed on demand in response to queries.	
Responsibilities:	**Collaborators:**
[2] To receive notifications from participants	⇒ Traders
[3] To respond to queries on market participants	⇐ Traders
External Interfaces:	
To store information on market participants	
Prerequisites: None	
Implementation: Modified version of default Facilitator; see Case Study #1	

Distributed Marketplace: Common Variations

The role model of Figure DM1.1 represents the 'lowest common denominator' of distributed marketplace applications. In practice this role model is likely to be extended through the addition of the following related roles:

Marketplace Visualiser

Often a marketplace has a means of visualising transactions. This is typically achieved by employing the Observer pattern, where the Visualiser is the observer and the other agents play the role of Subjects.



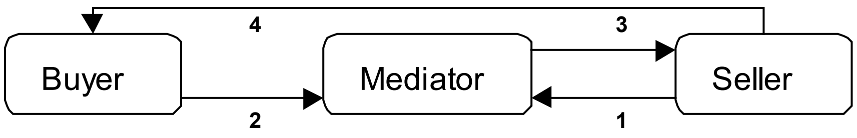
Interaction Summary:

	Collaboration	Explanation
1	Inform Request	Subjects are asked to forward activity reports
2	Activity Notification	Activity reports are dispatched

As the Zeus tool-kit provides a general purpose Visualiser agent, see the `Zeus Application` role model for its role description.

Transaction Mediator

In cases where the trading parties can not be relied upon to honour agreements a trusted third party is often used to facilitate payment and the transfer of goods. The mediator role is commonly found in conventional non-agent commerce, good examples being clearing organisations like Amex and Visa.



Interaction Summary:

	Collaboration	Explanation
1	Sale Notification	Terms of sale (e.g. price, item) sent
2	Payment Authorisation	Payment made to Mediator
3	Payment Notification	Payment verified, commission taken (if relevant), remainder transferred to Seller
4	Ownership Notification	Message confirming sale and transfer of goods

4.2 The Institutional Auction

ROLE MODEL

Abstract

This role model illustrates a centralised agent auction. The term *institution* has been deliberately chosen to convey the presence of local protocols and conventions that are enforced within this type of marketplace. In contrast to the decentralised market, trading in this model occurs indirectly using independent mediators. Hence an `Institutional Auction` involves a greater degree of regulation than the previously described `Distributed Marketplace`. For instance, admittance is subject to approval by an arbiter, bids are binding, and abuse and non-payment are subject to sanctions.

In the context of this role model an 'auction' is defined as a trading session mediated by an independent 'auctioneer', which chooses how the auction will be conducted. This is a significant point, as auctions can be run using a variety of rules. For instance, the stereotype of a fast talker calling out prices that increase as others bid is known as an 'English' or 'ascending-price open outcry' auction. Another protocol is where prices start high and descend until someone accepts: the 'Dutch' or 'descending-price open outcry' auction. An alternative to outcry auctions are sealed bid auctions, like the 'highest price, sealed bid' auction or the 'Vickrey' auction, where the second highest price wins. Some auction protocols work best for particular circumstances, and so the choice of which to use will often depend on who is buying and selling, what is being sold, and the timeframe involved.

As Exemplified by:

FISHMARKET	"Towards a Test-bed for Trading Agents in Electronic Auction Markets": Rodriguez-Aguilar et al, AI Communications 1999, In Press. http://www.iiia.csic.es/Projects/fishmarket/
AuctionBot	"The Michigan Internet AuctionBot": Wurman et al, Proceedings of the 2nd Int. Conference on Autonomous Agents 1998, p301-8. http://auction.eecs.umich.edu/

Similarities and Differences

This role model uses the name server role from the `Zeus Application` role model to provide network independence. As the identity of the auction institution could be encoded into the trading agents, a facilitator is not essential, but including one provides greater flexibility. Both roles are omitted from figure IA1.1 for clarity.

Role Model Diagram

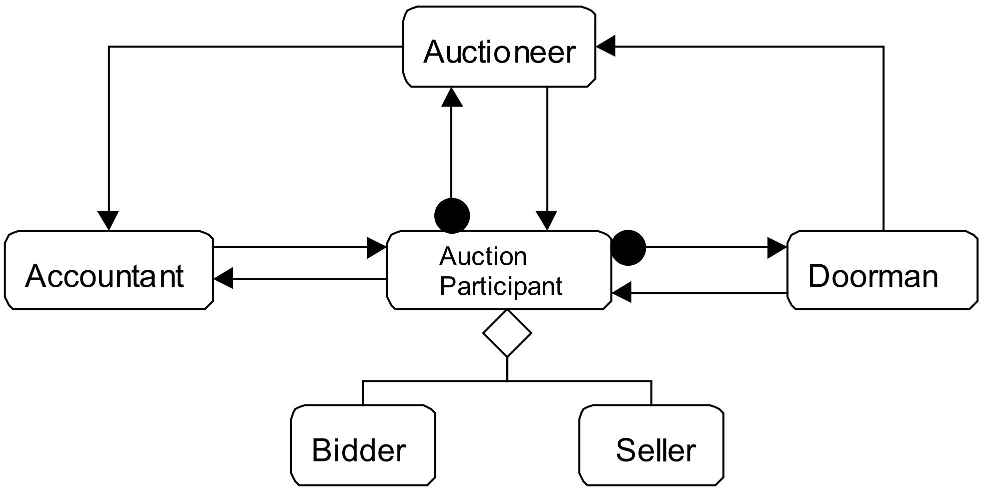


Figure IA1.1: The roles present in a typical Institutional Auction

Collaboration Diagram

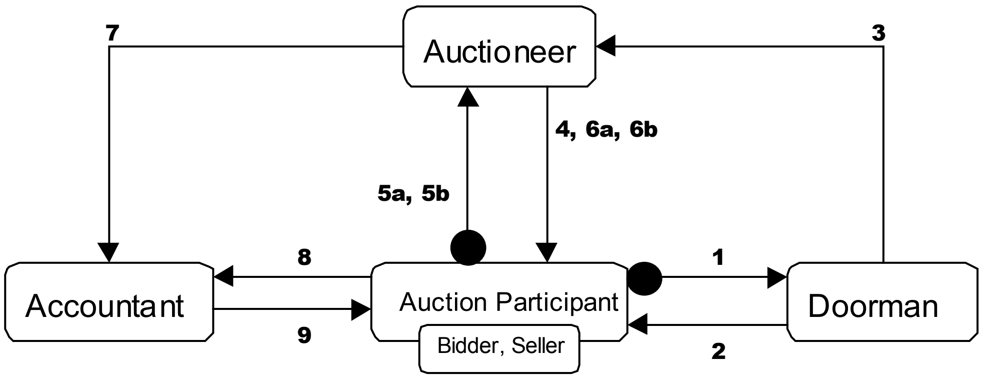


Figure IA1.2. The interactions between the roles of a Institutional Auction

Interaction Summary:

	Collaboration	Explanation
1	Admission Request	Doorman asked for permission to join the auction
2	Admission Response	Doorman notifies applicants
3	Member Notification	Auctioneer notified of changes in membership
4	Availability Notification	Participants notified as to what is on offer
5a	Bid	Bid sent to auctioneer
5b	Offer	Auctioneer asked to offer item for sale
6a	Bid Response	Bidders informed of auction status
6b	Offer Response	Sellers informed of their items' status
7	Sale Notification	Successful bid forwarded to Accountant
8	Settlement	Successful bidder makes payment
9	Ownership Transfer	Accountant notifies bidder of ownership transfer

Institutional Auction Role Descriptions

BIDDER	
Role Model: Institutional Auction	
Relationships to other Roles: Contained by the `Auction Participant` role	
Description: Those playing this role will be capable of proposing bids, either proactively or on behalf of their users. Proactive bidders will typically use bidding strategies, but this is not assumed.	
Responsibilities:	**Collaborators:**
[1] To request entry to / notify departure from auction	⇒ Doorman
[4] To receive information on items available for sale	⇐ Auctioneer
[5a] To submit bids	⇒ Auctioneer
[6a] To receive updates on auction status	⇐ Auctioneer
[8,9] To settle payment and receive bought items	⇐ ⇒ Accountant
External Interfaces:	
To facilitate the entry of user bids	
Prerequisites:	
Must possess a bidding protocol complementary to that used by the auctioneer	

SELLER	
Role Model: Institutional Auction	
Relationships to other Roles: Contained by the `Auction Participant` role	
Description: Those playing this role will be capable of offering items for sale. The strategy used to sell the item will typically be chosen according to institution rules, but some applications may allow sellers to nominate the means by which their items are sold.	
Responsibilities:	**Collaborators:**
[1] To request entry to / notify departure from auction	⇒ Doorman
[5b] To submit items for sale	⇒ Auctioneer
[6b] To receive updates on incoming bids	⇐ Auctioneer
[8] To receive payment on the sale of an item	⇐ Accountant
External Interfaces:	
To facilitate the entry of seller preferences	
Prerequisites:	
Requires a means of submitting an item for sale	

AUCTION PARTICIPANT
Role Model: Institutional Auction
Relationships to other Roles: specialises `Task Agent`, contains `Bidder` and `Seller`
Implementation: See Institutional Auction Case Study

DOORMAN	
Role Model: Institutional Auction	
Relationships to other Roles: specialises `Task Agent`	
Description: This role serves to regulate membership of an auction. Those attempting to join the auction must ask to be admitted, allowing the doorman to vet potential members. The doorman also informs the auctioneer of changes in membership as new members join and existing ones leave.	
Responsibilities:	**Collaborators:**
[1, 2] To process requests for entry to the auction	⇐ ⇒ Auction Participants
[3] To inform of changes in auction participants	⇒ Auctioneer
External Interfaces:	
To knowledge of auction rules, enables enforcement of institutional rules	
Prerequisites:	
Request, Receive and Notify protocols	

AUCTIONEER	
Role Model: Institutional Auction	
Relationships to other Roles: specialises `Task Agent`	
Description: The trading within an Institutional Auction is conducted through the player of this role. It receives instructions to sell, notifies potential buyers and processes their bids according to the auction protocol currently being employed. The auctioneer delegates settlement to another role: the accountant, which is notified at the same time as the winning bidder.	
Responsibilities:	**Collaborators:**
[3] To maintain a record of auction participants request	⇐ Doorman
[4] To open an auction and invite bids	⇒ Auction Participants
[5a] To receive incoming bids	⇐ Auction Participants
[5b] To receive requests to sell	⇐ Auction Participants
[6b] To inform seller of sale status	⇒ Auction Participants
[6a, 7] To award the winning bid	⇒ Accountant, Auction Participants
External Interfaces:	
To knowledge governing auction protocol selection	
Prerequisites:	
At least one auction protocol	

ACCOUNTANT

Role Model: Institutional Auction

Relationships to other Roles: specialises `Task Agent`

Description:
This role enables the settlement aspect of trading to be separated from the negotiation role of the auctioneer. Of course the auctioneer and accountant roles could be played by the same entity - but that is a design decision.

Responsibilities:	**Collaborators:**
[7] To receive notification of winning bids	⇐ Auctioneer
[8] To receive payment verification from buyer	⇐ Bidder
[9] To send notification of settlement	⇒ Bidder, Seller

External Interfaces:
To debit the account of the winning bidder
To calculate and bank the auctioneer commission
To credit the account of the appropriate seller
To transfer ownership of purchased items

Prerequisites:
Social protocol for notifying the transaction participants

5 THE BUSINESS PROCESSES DOMAIN

Constituent Role Models

1) Simple Supply Chain

2) Contractual Supply Chain (Variation)

Domain Characteristics

- Applications in this domain are characterised by entities whose continued operation is dependent on the consumption of resources produced by other entities.

- The role models of this domain describe the process by which resources are produced and consumed. Some are simple, releasing a resource when it is demanded, whilst more sophisticated variants involve some negotiation.

Business processes share many features with architectural software patterns, as both involve descriptions of participants, resources and interactions. Future releases of this document may use the wide range of business model literature that exists as inspiration for more business process role models.

Useful Links

To follow.

5.1 The Simple Supply Chain

Abstract

A Supply Chain has Suppliers who produce items and Consumers who consume them. A Consumer can have many Suppliers, but a Supplier only has one Consumer in any given Supply Chain. In its simplest form a Supply Chain exhibits three behaviours: planning, negotiation/commitment, and delivery.

At the highest level, a Supply Chain is made up of SC (Supply Chain) Predecessors and SC Successors. A Predecessor can have many Successors. As shown in the role model diagram in Figure SC1.1, a SC Participant is both a Predecessor and a Successor, while a SC Head is a refinement of a Predecessor, and a SC Tail refines a Successor.

As Exemplified by:

"Co-ordinating with Obligations", Barbuceanu et al, Proceedings of Autonomous Agents '98, May 1998, p. 62 - 69.
"Agent Enhanced Workflow", Judge et al, BT Technology Journal, July 1998, p79-85.

Similarities and Differences

This role model uses the name server role from the `Zeus Application` role model to provide network independence. Although the identity of resource producers could be encoded into the agents that will use them, a facilitator is recommended for flexibility. Both roles are omitted from figure SC1.1 for clarity.

In addition, all the SC roles are assumed to inherit the roles of the generic Task Agent, and so these are not listed here either.

Simple Supply Chain Role Model Diagram

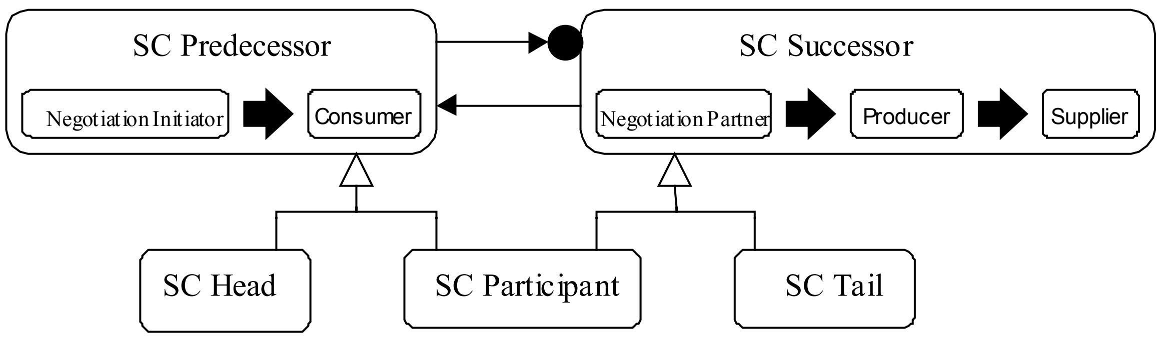


Figure SC1.1: The roles present within a Simple Supply Chain

Figure SC1.1 shows the negotiation and delivery phases of a supply chain as the sequence of roles played. For instance, the SC Predecessor goes from being a Negotiation Initiator to a Consumer, whilst a SC Successor is first a Negotiation Partner, then a Producer, and then finally a Supplier. These two roles are specialised into the SC Head, SC Participant and SC Tail roles.

Simple Supply Chain Collaboration Diagrams

During negotiation, the Negotiation Initiator role of the SC Predecessor negotiates with the Negotiation Partner in the Successor; the top half of Figure SC1.2 illustrates this interaction. If supply is agreed upon, the production and delivery of the resource in question follow, as shown in the bottom half of Figure SC1.2

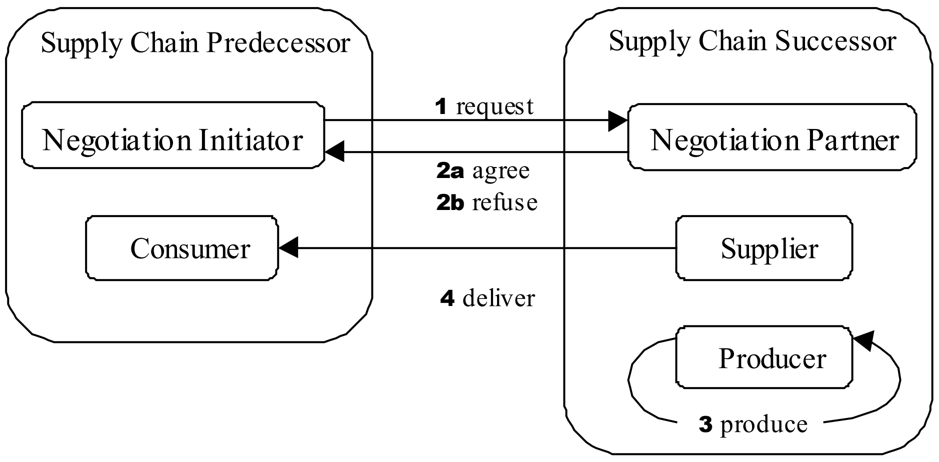


Figure SC1.2: The collaboration involved in initiating a supply chain and then delivering produced resources

This pattern is repeated for the length of the Supply Chain, as shown in Figure SC1.3. This diagram shows how the roles of SC Predecessor and SC Successor are specialised into SC Head, SC Participant and SC Tail according to their position within the supply chain. During negotiation, the pattern commences at the SC Head, and finishes at the Tail.

Once agreement is reached during negotiation, the delivery phase is entered, this proceeds in the opposite direction, as can be seen in Figure SC1.3, where production begins in the SC Tail and results in consumption in the SC Head. Another aspect to note in this diagram is that all interactions are numbered as decimals, this represents the fact that the SC Head and SC Participant 1 can not finalise their agreement until the entire chain has committed itself.

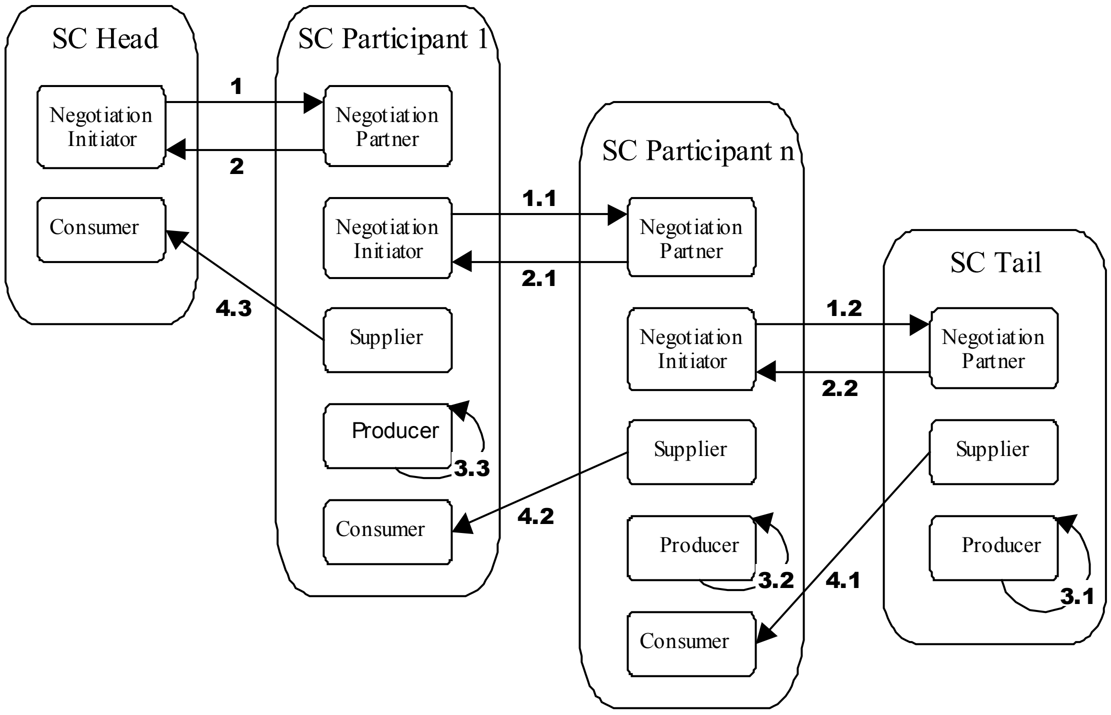


Figure SC1.3: The interactions between the roles in a typical supply chain

Interaction Summary:

	Collaboration	Explanation
1	Supply Request	Negotiation Initiator asks for resource
2	Supply Response	Negotiation Partner either accepts or refuses
3	Produce	If resource is produced externally, it is acquired
4	Deliver	Supplier transmits resource to Consumer

Note: The above interactions assume that the actions of provisioning and commitment are atomic, i.e. that an agent will wait until it knows if it can supply a resource before committing itself. If it were commit itself before being relatively certain of meeting its commitment, additional interactions would be necessary for failure conditions.

Other variations in the collaboration sequence are common. For example, any node of the chain can unexpectedly fail during production, and a contractual arrangement can be terminated due to poor performance or other issues.

Simple Supply Chain Role Descriptions

Supply Chain Head :: Negotiation Initiator	
Role Model: Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: This is the role responsible for negotiating delivery of a resource on the behalf of the supply chain head. As the SC Head role does not contain a Producer sub-role it is meant as a gateway into a business process, i.e. one that will trigger the production of resources, but not locally.	
Responsibilities:	**Collaborators:**
[1,2] To negotiate terms of supplying a resource	⇐ ⇒ Negotiation Partner
[3] To pass information on expected resources	⇒ (next own role) Consumer
External Interfaces:	
To negotiation expertise [optional]	
Prerequisites:	
Will be aware of at least one contract initiation protocol	

Supply Chain Head :: Consumer	
Role Model: Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: This sub-role receives resources requested by the Negotiation Initiator, and then consumes them (the manner in which the resources are used is applicant dependent). If the resources are to be paid for, this role should also handle settlement.	
Responsibilities:	**Collaborators:**
[4] To receive resource deliveries	⇐ Supplier
External Interfaces:	
To payment settlement system [optional]	
To domain specific consumption activity	
Prerequisites: None	

Supply Chain Participant :: Negotiation Partner	
Role Model: Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: This role receives resource requests from entities that suspect it will be able to satisfy their requirements. The SC Participant role is different from the SC Tail role because it will need additional resources to fulfil these incoming requests, thus this role will become a Negotiation Initiator. Once the delivery of the necessary resources has been negotiated the SC Participant will move into the Consumer role and await their arrival.	
Responsibilities:	**Collaborators:**
[1,2] To negotiate supply of resources	⇐ ⇒ Negotiation Initiator
To ask for supplies	⇒ (next own role) Negotiation Initiator
To pass necessary resources	⇒ (next own role) Consumer
External Interfaces:	
To negotiation expertise [optional]	
Prerequisites:	
Will be aware of at least one contract respondent protocol	

Supply Chain Participant :: Negotiation Initiator	
Role Model: Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: This is the sub-role played by SC Participants who are seeking particular resources. This role only concerns negotiation for resources, delivery being handled by the Negotiation Partner role.	
Responsibilities:	**Collaborators:**
[1] To request necessary resources	⇒ Negotiation Partner
[2] To receive notification on previous requests	⇐ Negotiation Partner
To pass previously accepted resource request	⇒ (next own role) Negotiation Partner
External Interfaces:	
To negotiation expertise [optional]	
Prerequisites: Will be aware of at least one contract initiator protocol	

Supply Chain Participant :: Producer **Supply Chain Tail :: Producer**	
Role Model: Supply Chain	
Relationships to other Roles: Specialises `Task Agent`	
Description: This sub-role is common to both SC Participants and SC Tails, it encapsulates some means of production, (this is application dependent), and forwards the results to the Supplier sub-role.	
Responsibilities:	**Collaborators:**
To pass produced resource	⇒ (next own role) Supplier
External Interfaces:	
To means of production of resources	
Prerequisites: None	

Supply Chain Participant :: Supplier **Supply Chain Tail :: Supplier**	
Role Model: Supply Chain	
Relationships to other Roles: Specialises `Task Agent`	
Description: This sub-role is an intermediary, receiving the resources from the Producer role, and delivering them as previously agreed to the appropriate consumer.	
Responsibilities:	**Collaborators:**
To receive newly created resources	⇐ (own) Producer
[4] To deliver appropriate resources	⇒ Consumer
External Interfaces: None Assumed	
Prerequisites: None	

Supply Chain Participant :: Consumer	
Role Model: Supply Chain	
Relationships to other Roles: Specialises `Task Agent`	
Description: This sub-role is an intermediary, accepting delivery of needed resources, and forwarding them to the Producer sub-role.	
Responsibilities:	**Collaborators:**
[4] To accept delivery of resources	⇐ Supplier
To pass on received resources	⇒ (next own role) Producer
External Interfaces:	
To domain-specific consumption activity	
To payment settlement system [optional]	
Prerequisites: None	

Supply Chain Tail :: Negotiation Partner	
Role Model: Supply Chain	
Relationships to other Roles: Specialises `Task Agent`	
Description: This role receives resource requests from entities that suspect it will be able to satisfy their requirements. However, unlike like the equivalent sub-role of SC Participants, this sub-role requires no additional resources to the request, enabling it to immediately either agree or reject the opportunity to supply. If it agrees to supply, it will move into the Producer role.	
Responsibilities:	**Collaborators:**
[1] To receive resource requests	⇐ Negotiation Initiator
[2] To respond (accept/reject) resource requests	⇒ Negotiation Initiator
To pass accepted request	⇒ (next own role) Producer
External Interfaces:	
To negotiation expertise [optional]	
Prerequisites: Will be aware of at least one contract respondent protocol	

5.2 The Contractual Supply Chain Variation

Abstract

The negotiation in the Simple Supply Chain (cf.) only allows a request to be accepted or rejected. Often however competitive bidding would be used to tender the production of necessary resources. This role model illustrates how a Contract Net can be used for the negotiation phase, using the negotiation roles in the Contract Net to replace the Negotiation Initiator and the Negotiation Partner roles in the Simple Supply Chain. The delivery process is unchanged.

As Exemplified by:

"Agent Standards", I. Dickinson, FIPA Agent Technology Group, 1997. Http://drogo.cselt.stet.it/fipa

Similarities and Differences

This role model uses the name server role from the `Zeus Application` role model to provide network independence. Although the identity of resource producers could be encoded into the agents that will use them, a facilitator is recommended for flexibility. Both roles are omitted from figure CSC1.1 for clarity.

Contractual Supply Chain Role Model Diagram

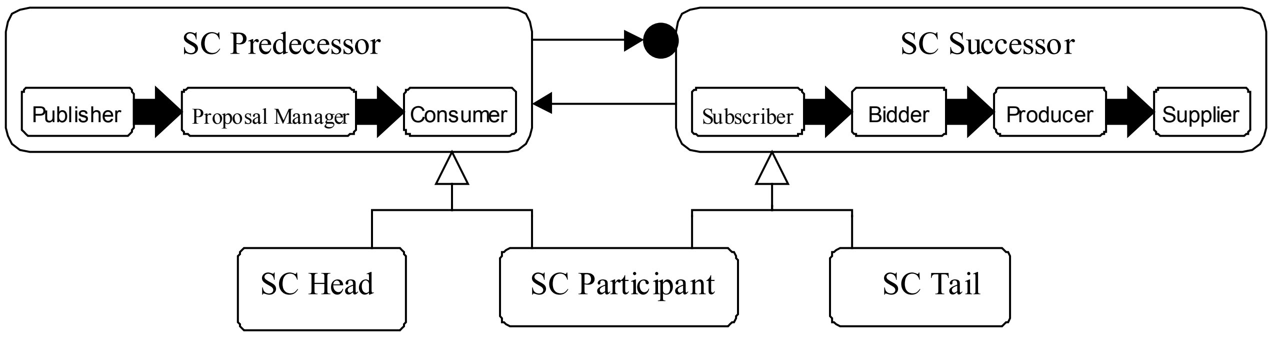


Figure CSC1.1: The roles present within a Simple Supply Chain

Figure CSC1.1 shows the negotiation and delivery phases of a Contractual Supply Chain as the sequence of roles played. In contrast to the Simple Supply Chain (q.v.) when negotiation consisted of a single Negotiation Initiator role, the SC Predecessor has two roles during negotiation, Publisher and then Proposal Manager. These are complemented by the Subscriber and Bidder roles of SC Successor. Like the Simple Supply Chain these two roles are specialised into the SC Head, SC Participant and SC Tail roles.

Contractual Supply Chain Collaboration Diagrams

During negotiation, the Negotiation Initiator role of the SC Predecessor negotiates with the Negotiation Partner in the Successor; the top half of Figure SC1.2 illustrates this interaction. If supply is agreed upon, the production and delivery of the resource in question follow, as shown in the bottom half of Figure SC1.2.

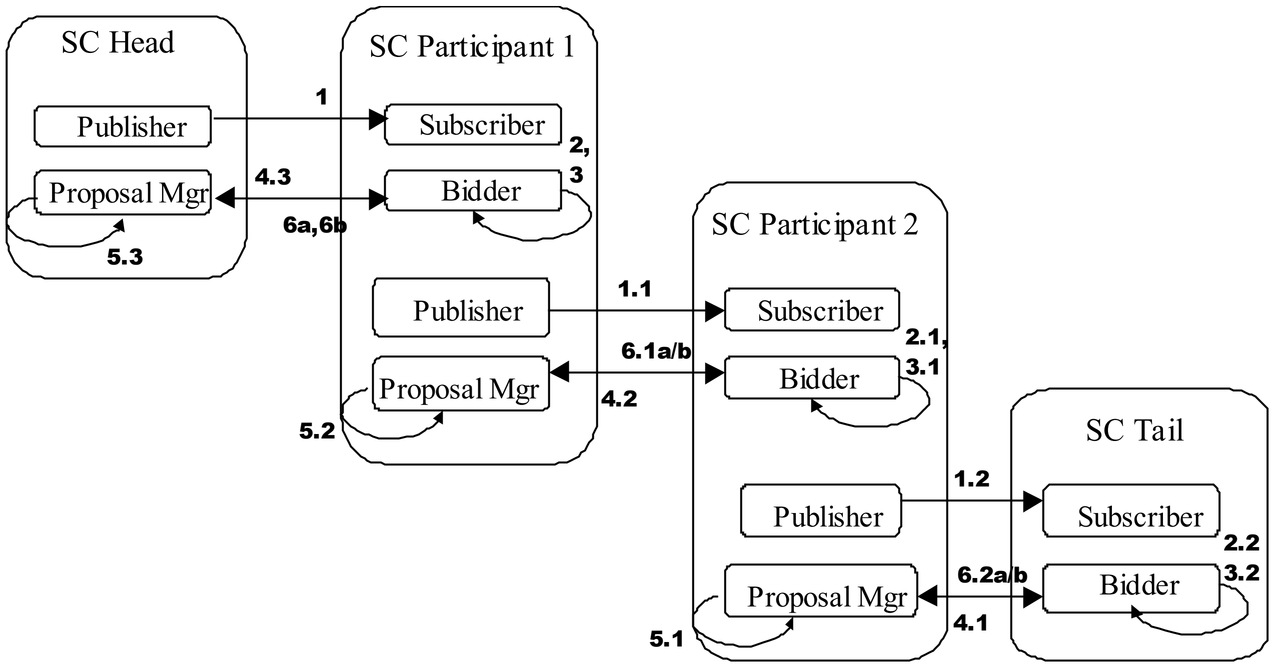


Figure CSC1.2: The interactions between the roles during the Negotiation phase of a typical Contractual Supply Chain

Interaction Summary:

	Collaboration	**Explanation**
1	CFP Published	Call for Proposals issued
2	Consider Proposal	Subscribers consider whether to bid
3	Bid Formulated	Bid planned and priced
4	Bid Issued	Bid is sent to proposer
5	Bid Consider	Received bids are evaluated
6a	Bid Accepted	Originator is informed, bid enacted
6b	Bid Rejected	Originator informed bid has not been successful

The collaboration diagram Figure CSC1.2 assumes that a Bidder will always agree to the terms of an accepted bid, giving all Proposal Managers total control. In that case, once the Bidder role in a SC Participant has their bid accepted or rejected, they notify their own Proposal Manager to proceed with accepting and rejecting the bids that have been received from their SC Successor. Once the Proposal Manager in a SC Participant has completed this step, they pass control to their own Consumer role to await delivery from their SC Successor.

The above interactions also assume that the actions of provisioning and commitment are atomic, i.e. that an agent will wait until it knows if it can supply a resource before committing itself.

Contractual Supply Chain Role Descriptions

Publisher - Supply Chain Head

Role Model: Contractual Supply Chain

Relationships to other Roles: Specialises `Task Agent`

Description:
This is the role responsible for issuing proposals to all interested parties. If proposals are to be sent selectively a registry of listener interests is necessary, although such information can be obtained from an agent in the Facilitator role (q.v.), if one exists.

Responsibilities:	**Collaborators:**
[1] To publish new proposals	⇒ Subscribers
To pass on issued CFPs	⇒ (next own role) Proposal Manager

External Interfaces:

To application-specific component that triggers the CFP

Prerequisites: Will be aware of at least one contract initiator protocol

Proposal Manager - Supply Chain Head

Role Model: Contractual Supply Chain

Relationships to other Roles: Specialises `Task Agent`

Description:
This role is responsible for processing incoming bids, which it can either accept or reject. In the event of an accepted bid this role moves into the consumer role to await delivery of the agreed resources.

Responsibilities:	**Collaborators:**
[6a] To accept successful bids	⇒ Bidders
[6b] To reject unwanted bids	⇒ Bidders
To pass accepted proposals	⇒ (next own role) Consumer

External Interfaces:

To bid evaluation expertise to consider received bids

Prerequisites: Will be aware of at least one contract respondent protocol

Consumer - Supply Chain Head

See Role Description in Simple Supply Chain

Subscriber - (SC Participant, SC Tail)	
Role Model: Contractual Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: This role receives 'Calls For Proposals' (CFPs): invitations to bid for a particular resource, usually given some constraints. To receive such invitations the Subscriber must be known to the CFP originator, either by previously registering with it, or by being known by an intermediary like a Facilitator (q.v.). Once a CFP is received the role of Bidder is assumed.	
Responsibilities:	**Collaborators:**
[1] To receive CFP notifications	⇐ Publishers
To register with directory service	⇒ Facilitator
To pass received proposals	⇒ (next own role) Bidder
External Interfaces: None Assumed	
Prerequisites: None	

Bidder - Supply Chain Participant	
Role Model: Contractual Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: The role of the Bidder is to evaluate received CFPs and formulate an appropriate response. SC Participants differ from SC Tails by the need to tentatively secure additional resources before a bid of their own can be issued; hence this will assume the role of Publisher and issue its own CFP before replying to the original invitation. If the resulting bid is accepted the Bidder will then assume the Proposal Manager role to notify its subcontractors.	
Responsibilities:	**Collaborators:**
[4] To respond (send bid / reject proposal)	⇒ Proposal Manager
To ask for proposals from suppliers	⇒ (next own role) Publisher
To pass accepted bid	⇒ (next own role) Proposal Manager
External Interfaces:	
To expertise that will consider CFPs	
To expertise that will formulate bids	
Prerequisites: Will be aware of at least one contract respondent protocol	

Producer - (SC Participant, SC Tail)
See Role Description in Simple Supply Chain

Proposal Manager - Supply Chain Participant	
Role Model: Contractual Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: This role enables SC Participants to subcontract resource production. During the negotiation phase it will pass bids received from subcontractors (i.e. its successors in the supply chain) to its own Bidder role, which will evaluate the responses and consider its own bid to the CFP originator. During the delivery phase it will wait for an agreed resource to be delivered before assuming the Consumer role and processing it.	
Responsibilities:	**Collaborators:**
[6a] To notify bid acceptance	⇒ Bidders
[6b] To notify bid rejection	⇒ Bidders
To pass bids from Successors	⇒ (next own role) Bidder
To pass previously accepted proposal	⇒ (next own role) Consumer
External Interfaces:	
To evaluation expertise, to consider received bids	
Prerequisites: None	

Supplier - (SC Participant, SC Tail)
See Role Description in Simple Supply Chain

Bidder - Supply Chain Tail	
Role Model: Contractual Supply Chain	
Relationships to other Roles: Specialises the `Task Agent` role	
Description: The SC Tail variant of the Bidder role differs in that does not attempt to subcontract CFPs, it must formulate its bid on the basis of whether or not it can fulfil the requirements.	
Responsibilities:	**Collaborators:**
[4] To respond (send bid / reject proposal)	⇒ Proposal Manager
To pass accepted bid	⇒ (next own role) Producer
External Interfaces:	
To expertise that will consider CFPs	
To expertise that will formulate bids	
Prerequisites: Will be aware of at least one contract respondent protocol	

FUTURE WORK

The role models in this document are just a sample of a wide range of roles to be found amongst current distributed applications. In an effort to extract and document additional role models we are currently studying several other domains, such as:

- Monitoring and Control, e.g. telecommunications network management
- Information Management, e.g. information mining and distribution
- Collaborative Analysis
- Distributed Simulation
- Manufacturing and Logistical Management

The results of our role mining will be described in future releases of this document, available from our WWW home page. One domain of particular interest to our group is that of Interface agents: software that provides a smarter interface for its users. The work described in [6] has already partitioned this domain and identified the following roles:

Interface Agents that Serve Individuals	**Interface Agents that Serve Communities**
Role: Assistant An application specific guide	Role: Matchmaker A social broker that finds those with some criteria
Role: Critic, Continuity Advisor Offers an evaluation service	Role: Librarian Typically a Finding and Cataloguing Service
Intelligent Gateway Also known as a Federated Database	
Information Screen A context or profile based filtering service	
Reporter, Analyst A monitoring and presentation service	
Oracle/ Consultant Provides an intelligent source of expertise	

Ultimately our intention is to document these roles and then describe how to best realise them using the ZEUS tool-kit.

In the meantime, help in locating and describing agent role models in as many diverse domains as possible is welcomed.

Jaron Collis (jaron@info.bt.co.uk)

REFERENCES

[1] Kendall, E. A. *Agent Roles and Role Models.* Available from: http://www.labs.bt.com/projects/ibsr/papers/patterns/iaipm.ps.gz

[2] Kristensen, B. B., “Object-Oriented Modelling with Roles”, *OOIS'95, Proceedings of the 2nd International Conference on Object-Oriented Information Systems*, 1996. http://www.mip.ou.dk/~bbk/research/recent_publications.html.

[3] Rational Software. UML Documentation Resource Page http://www.rational.com/uml/resources/documentation/index.jtmpl

[4] Nwana, H.S. *Software Agents: An Overview*. The Knowledge Engineering Review, 11(3), p205–244, 1996. Available from: http://www.labs.bt.com/projects/agents/publish/

[5] Jennings N.R. & Wooldridge M. *Software Agents*, IEE Review, p17–20, January 1996. Available from: ftp://ftp.elec.qmw.ac.uk/pub/isag/distributed-ai/publications/IEE-Review96.ps.gz

[6] Masterton S. & Watt S., "Oracles, Bards, and Village Gossips, or Agents, Roles, and Meta-Knowledge Management", should soon be available through http://kmi.open.ac.uk/people/snw2/